INVESTIGATION OF NON-PHYSIOLOGICAL PRESSURE SOURCES ASSOCIATED WITH EXERCISE DYNAMICS PRESENT IN EQUINE PULMONARY ARTERY AND ESOPHAGEAL PRESSURES RECORDED WITH A MILLAR TRANSDUCER

by

DEANNA L. CARROLL

B.S., Kansas State University, 1986

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Approved by:

Richard R. Gallagher
Major Professor

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

CHAPTER PAGE

LIST OF FIGURES

LIST OF FIGURES (cont.)

LIST OF TABLES

I. INTRODUCTION

The high incidence of pulmonary hemorrhage in exercising horses has led to several studies investigating pulmonary arterial pressure. Evidence suggests that a causal link between pulmonary hemorrhage and exercise exists [3]. Much of the pulmonary hemorrhage research has involved recorded pulmonary arterial and esophageal pressures under exercise conditions. This research investigated the signal components of exercise and post-exercise pulmonary arterial and esophageal pressures recorded with Millar transducers. The primary research objectives were 1) to demonstrate that a Millar pressure transducer, in addition to recording physiological pressures, records pressures due to non-physiological sources associated with the dynamics of exercise and 2) to develop a procedure for identifying and isolating the components of a pressure signal due to non-physiological sources.

II. SIMPLE PULMONARY ARTERY MODEL

2.1 Experimental Purpose

A simple model of the equine pulmonary artery was constructed for investigation of non-physiological pressure components included in the recordings from a Millar pressure transducer during equine exercise studies. Anticipated pressure sources include hoof impact, acceleration forces due to movements of structures within the abdominal and chest cavities, transducer impact against artery walls, and transducer movement within the blood vessel itself.

2.2 Experimental Apparatus and Procedure

A balloon, chosen for its elastic properties, was filled with water to the approximate dimensions of 4 cm diameter x 20 cm length. A Millar (Model SPC-360(B), Millar Instruments, Houston, TX) catheter pressure transducer was inserted into the balloon. The balloon opening was sealed around the catheter using a rubber band. The output of the pressure transducer control unit (Model TC-500D, Millar Instruments, Houston, TX) was amplified using a precision instrumentation amplifier (AD521J Analog Devices, Norwood, MA) with a programmable gain of 100 and then displayed on an oscilloscope (Model 5111A, Tektronix, Inc; Beaverton, OR). A schematic of the amplifier circuit is shown in Fig. 2.1.

System calibration was performed using a mercury manometer and the 100 mmHg pressure signal generated by the

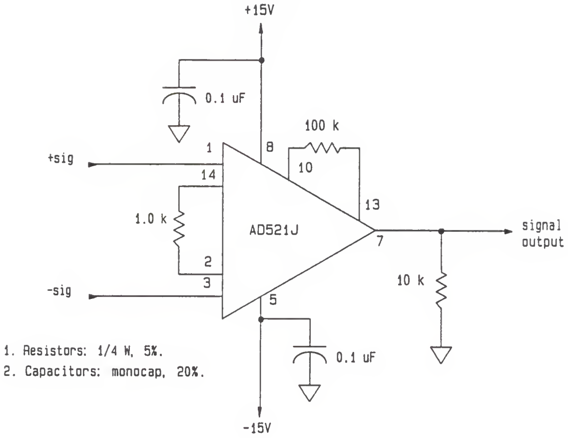


Figure 2.1. Amplifier Circuit for Pulmonary Artery Model. A precision instrumentation amplifier was used to amplify the output of a pressure transducer control unit before displaying it on an oscilloscope.

control box. The following tests were conducted for a variety of catheter insertion distances:

a) The balloon was vibrated.

b) Small pressure impulses were applied to the balloon wall.

c) The catheter line was vibrated.

d) The catheter line was vibrated with sufficient force to cause the transducer tip and the transducer side to hit the balloon wall.

After removal from the balloon, the catheter was inserted into a bucket filled with water. The following tests were conducted using this rigid model:

a) The catheter line was vibrated.

b) The transducer tip was tapped against the bucket wall.

2.3 Experimental Observations

Due to the simplicity of the model, precise results were not possible; however, general observations were useful. Each test performed resulted in the oscilloscope display signal changing from a constant value, due to the hydrostatic pressure, to a signal comprised of numerous transients. The magnitude of the pressure transients varied; however, a pressure change of ±5 mmHg was common.

2.4 Discussion

The experiments showed that the model was sensitive to externally applied inputs, possibly resulting in a pressure signal related to inertial effects. Thus, a hypothesis was formed:

Hypothesis: The Millar pressure transducer will record non-physiological pressures in addition to physiological pressures during conditions of exercise or rapid movements by the subject.

These experiments illustrated the difficulty of modeling the equine system with the necessary accuracy to obtain quantitative results. Further research with more extensive modeling of the pulmonary artery was not considered feasible, because of the numerous variables associated with a reasonably accurate model. Examples of these variables might involve exercise simulation, the damping effect of bone and tissue, and artery suspension. The difficulty of determining these variables led to consideration of studies using experimental data. Research using experimental data was determined to be practical since a reliable data collection system and analysis techniques were available.

III. EXERCISE/POST-EXERCISE EQUINE STUDIES

3.1 Experimental Purpose

To show that Millar transducer recordings of pulmonary arterial and esophageal pressures in exercising horses included non-physiological pressures, a comparison was made of data collected during exercise with data collected immediately upon termination of exercise, i.e., when the horse became stationary. It was assumed that physiological pressures associated with the cardiac cycle and respiration would remain essentially constant over a short time period following exercise; thus, differences between exercise and immediately post-exercise data windows would be due to the dynamics of exercise. Although the cardiac cycle and respiration did not remain constant throughout the stationary data window, observations were still possible. In addition, to demonstrate the significance of the signal component due to exercise, the dynamic pressure signal was filtered, eliminating the particular frequency components which were determined to be associated with exercise.

3.2 Experimental Apparatus and Procedure

The horse used in this experiment had previously been trained to run on a high-speed treadmill (SATO Inc., Uppsalla, Sweden) and to stand quietly before and after exercise. The horse was outfitted with a safety harness connected to an emergency shut-off switch located on the

treadmill.

Self-adhesive electrocardiograph electrodes (Quinton, Seattle, WA) were placed on the forehead, withers, and sternum. A 7F introducer catheter (USCI Cardiology and Radiology Division, CR Bard, Billerica, MS.) was placed in the right jugular vein. A catheter pressure transducer (Model SPC-761(P), Millar Instruments, Houston, TX) was introduced through the right jugular vein introducer and positioned in the pulmonary artery approximately 7 - 10 cm from the semilunar valve.

A second catheter pressure transducer (Model SPC-360(B), Millar Instruments, Houston, TX) was encased in polyethylene tubing and passed through the nose into the esophagus using a 2 cm (outside diameter) split tygon stomach tube. The transducer was positioned approximately 15 cm from the entrance to the stomach.

Pulmonary arterial pressure, esophageal pressure, and an ECG were recorded using three channels of a multichannel pen recorder (Model R-611, Beckman Instruments). Pulmonary arterial pressure and esophageal pressure were also recorded using two channels of an instrumentation tape recorder (Model HP3960, Hewlett-Packard). A block diagram of the experimental apparatus is shown in Fig. 3.1.

The treadmill speed was gradually increased to 11 m/sec. After approximately 20 seconds of exercise at 11 m/sec, the treadmill power was disconnected. Data were

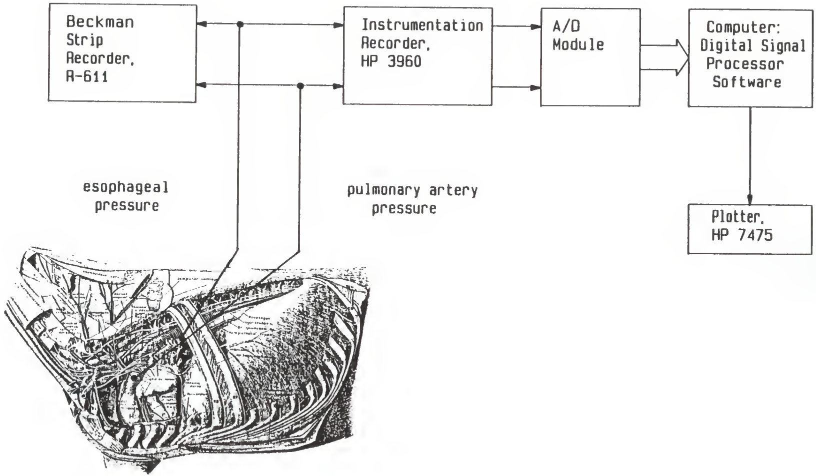


Figure 3.1. Exercise/Post-Exercise Experimental Apparatus. The instrumentation which was used to collect and analyze experimental data for this research.

recorded from the onset of exercise to 3 minutes post-exercise.

3.3 Experimental Data

The multichannel pen recording of pulmonary arterial pressure, esophageal pressure, and ECG is shown in Fig. 3.2. Analysis of the esophageal pressures and pulmonary arterial pressures are discussed in Sections 3.4.1 - 3.4.4. Calculations of heart rate from the ECG signal are discussed in Section 3.4.5.

3.4 Data Analysis

Analysis techniques allowed specific signal components of the pulmonary arterial and esophageal pressures to be identified and removed. These techniques and accompanying results are presented here. A guide for using the data analysis software is included in Appendix A.

3.4.1 Data Windows

A total of four data windows were collected from the recorded pressures. Pulmonary arterial and esophageal pressures were sampled for 20 seconds immediately prior to shutting off the treadmill power and for 20 seconds immediately following the time at which the horse became stationary. As shown in Fig. 3.2, approximately 2 seconds were necessary for the horse to become stationary after the power to the treadmill was terminated. A time line of the events is shown in Figure 3.3.

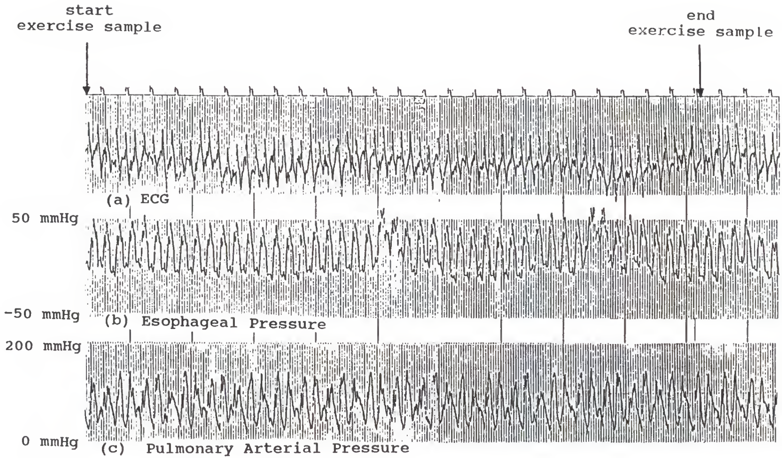


Figure 3.2. Exercise/Post-Exercise Experimental Data. Comparison with digitized data indicates no notable distortion was introduced by digitizing.

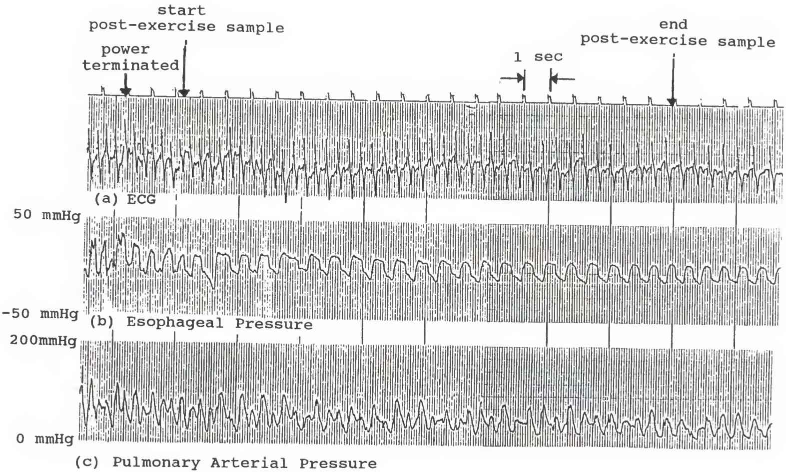


Figure 3.2. (cont.) Exercise/Post-Exercise Experimental Data.

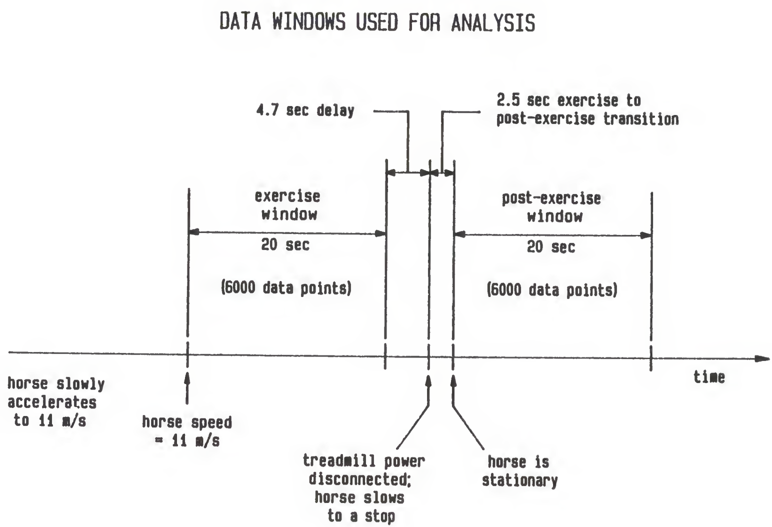


Figure 3.3. Data Windows Used for Analysis. A total of four windows were collected from the recorded pressures for analysis.

3.4.2 Digital Conversion

The recorded pressure signals were digitized using a modified data acquisition module [4, 7, 10]. The data acquisition module was a four channel, 12-bit, adjustable gain system which simultaneously sampled the channels.

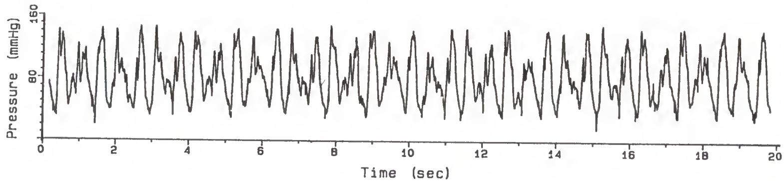
Each channel was calibrated using prerecorded pressure constants. Table 3.1 displays the calibration values and

Table 3.1: Calibration Values for the Data Acquisition System.

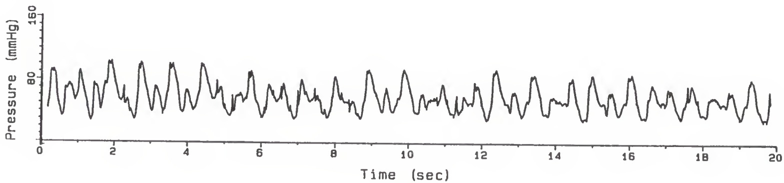
Type of signal	pressure (mmHg)	digital representation
pulmonary artery pressure	0 100	* 1978-2015
esophageal pressure	-50 0 50	416-453 1823-1857 3236-3286

* These readings were inconsistent. Values from the pen recording and the corresponding digital values were used to compute the pulmonary artery pressure conversion.

the corresponding digital values, which were used to convert the data to pressures to mmHg after analysis. Channel gains were adjusted to maximize the signal within the 12-bit range. A sampling frequency of 300 Hz was used, thus 6000 points were collected for each 20 second data window. Figs. 3.4 - 3.5 display the sampled data. The similarity of the sampled data and the data recorded with



(a)



(b)

Figure 3.4. Pulmonary Arterial Pressure Data (a) Exercise (b) Post-Exercise. Comparison with experimental data indicates no notable distortion was introduced by digitizing.

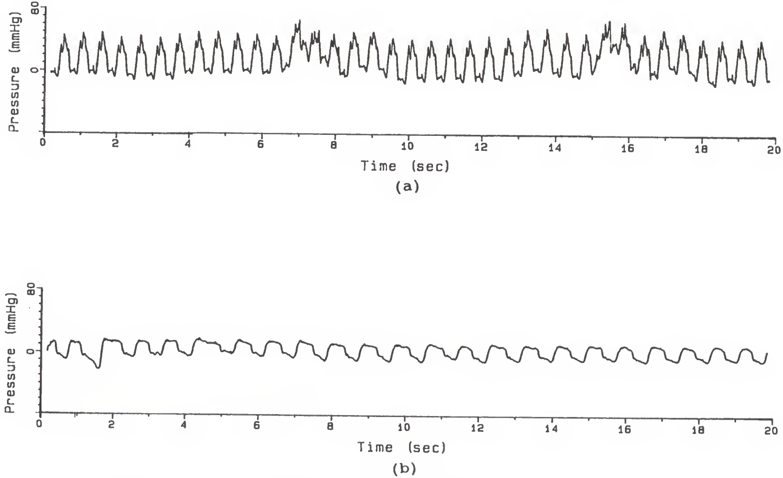


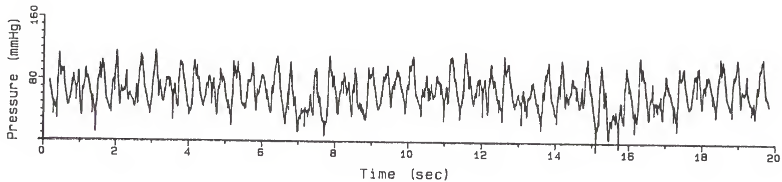
Figure 3.5. Esophageal Pressure Data (a) Exercise (b) Post-Exercise. Comparison with experimental data indicates no notable distortion was introduced by digitizing.

the multichannel pen recorder (Fig. 3.2) demonstrated that digitizing did not introduce notable distortion. A comparison of section A of Fig. 3.4 with section A of Fig. 3.2 demonstrated that low and high frequency signal content was reproduced reliably. Signal peaks and notches occur at the same time and the signal magnitudes are equivalent.

A digital signal processing software package [5] was used for data analysis. Input routines allowed signals to be retrieved from disk and also allowed the data to be plotted or printed on a terminal. The package offered several data processing options; this research required the autocorrelation and power spectral density algorithms. Processing results could be printed, plotted, or saved on disk.

3.4.3 Transmural Pulmonary Artery Pressure Windows

Excessive pressure across a vessel wall leads to its rupture causing hemorrhage. Thus, the pressure across the pulmonary artery wall, transmural pulmonary artery pressure, is of interest in pulmonary hemorrhage studies. Exercise and post-exercise transmural pulmonary artery pressures, shown in Fig. 3.6, were generated by mathematically subtracting the esophageal pressure from the pulmonary arterial pressure.



(a)

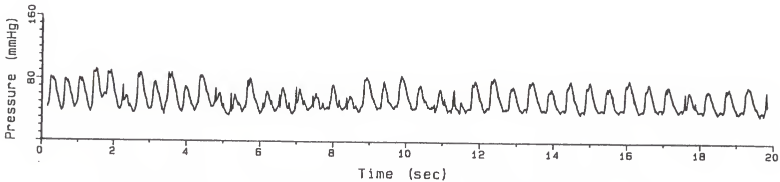


Figure 3.6. Transmural Pulmonary Artery Pressure Data (a) Exercise (b) Post-Exercise. Generated by mathematically subtracting the esophageal pressure from the pulmonary arterial pressure.

3.4.4 Power Spectral Density Estimation

In order to identify the frequency components present in the recorded pressures power spectral densities were estimated. PSD plots result from a method of describing a signal using frequency-domain concepts which involve Fourier transforms (FTs). The exact FT (therefore the exact PSD) of a signal cannot be determined because no knowledge exists of the data sequence beyond the sampling period; however estimations are possible. Since each window of data consisted of a large number of points, 6000, the PSDs were estimated using the direct method, which is illustrated in Fig. 3.7.

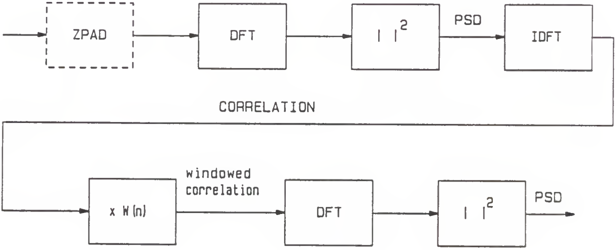


Figure 3.7. Power Spectral Density Estimation by Direct Method. Data is zero-padded to the nearest 2^n points. A PSD estimate is found as the squared magnitude of the discrete fourier transform. The correlation, found as the inverse discrete Fourier transform of the PSD was multiplied by a window function. A smooth PSD estimate was then obtained with the DFT of the windowed function.

The PSD estimation required 2^n data points, where n is any positive integer, so 0's were added to the data sequence until each data window consisted of 2^{13}, 8192, data points. The first estimate of the PSD, found as the squared magnitude of the discrete Fourier transform (DFT) of the sequence, was not used because of the variance in the estimate. The correlation, found as the inverse discrete Fourier transform (IDFT) of the PSD, was multiplied by the rectangular window function. Windowing is a technique which reduces the error associated with the FT estimation. Several window functions are commonly used. The rectangular window function was chosen because it is helpful in the detection of two components that are close together in frequency and have relatively equal amplitudes. A smooth PSD estimate was then obtained with the DFT of the windowed correlation. To eliminate magnitude changes due to diminishing dc-pressure the PSD magnitudes were normalized before plotting. This was accomplished by dividing the PSD values by the dc value which made the dc signal power 0 dB. Since the dc value was unique to each particular signal, normalization values changed between signals. Thus, direct comparisons of signal power between windows were not possible; however, comparisons of percentages of the total signal power present at a specific frequency were possible. All pressure signals had a large dc-offset compared to the

peak-to-peak pressures, thus the PSDs showed a large percentage of the signal power at 0 Hz. The PSDs of the six data windows are shown in Figs. 3.8 - 3.10.

3.4.5 Digital Filter Processing

To demonstrate the influence of the signal component associated with exercise dynamics the component was filtered from the recorded data. The data windows corresponding to exercise conditions were processed by a digital filter, removing the 1.88 Hz fundamental and its 2nd - 9th harmonics. Fig. 3.11 shows the filter's magnitude response. The digital filter was comprised of a series of stopband elliptic filters. An elliptic transfer function was selected for its superior low frequency stopband characteristics. Due to filter sensitivity, a 4-pole filter was required for the low frequency fundamental and 8-pole filters were required for the harmonics. The 4-pole filter coefficients were obtained from an elliptic approximation table [2]. An elliptic estimation routine and a graph of passband loss vs. selectivity factor were used to obtain the 8-pole filter coefficients [1]. The filter coefficients are listed in Appendix C. Fig. 3.12 illustrates the ladder network used in the filter implementation [9]. Listings of the filter design and implementation routines are included in Appendix B.

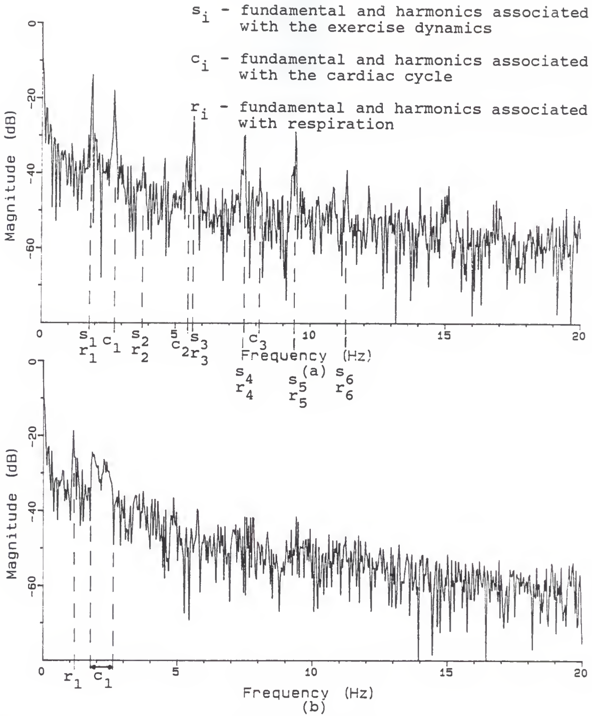


Figure 3.8. Pulmonary Arterial Pressure Power Spectral Density (a) Exercise (b) Post-Exercise. Signal component associated with exercise dynamics, s, (1.88 Hz) has larger percentage of total signal power during exercise than post-exercise.

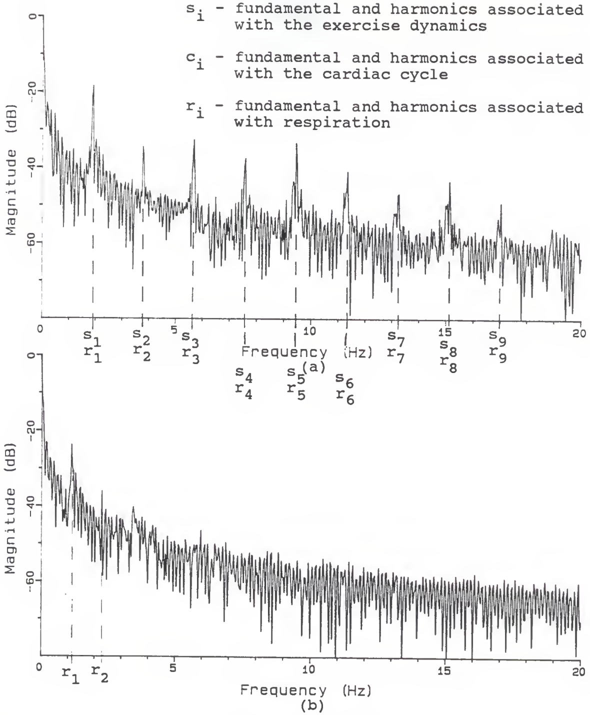


Figure 3.9. Esophageal Pressure Power Spectral Density (a) Exercise (b) Post-Exercise. The harmonics of the signal component associated with exercise dynamics, s, have a larger percentage of total signal power during exercise than post-exercise.

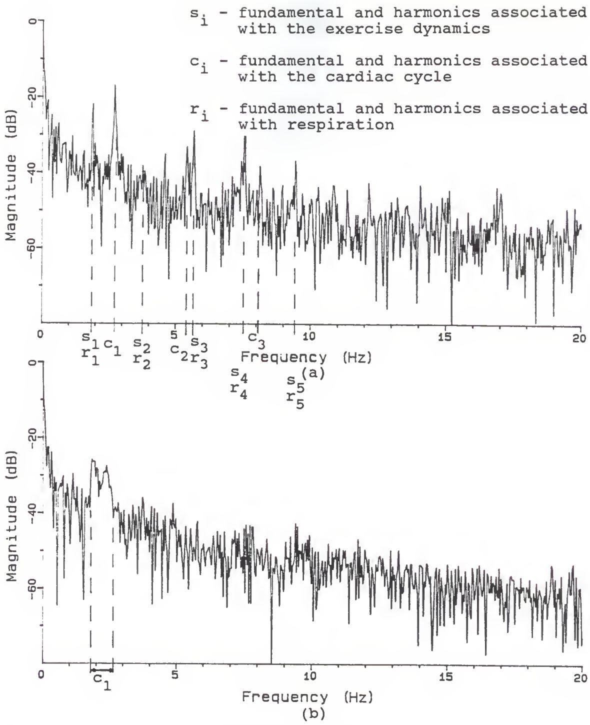


Figure 3.10. Pulmonary Arterial Transmural Pressure Power Spectral Density (a) Exercise (b) Post-Exercise. Signal component with 1.88 Hz fundamental present in exercise but not in post-exercise, due entirely to exercise dynamics.

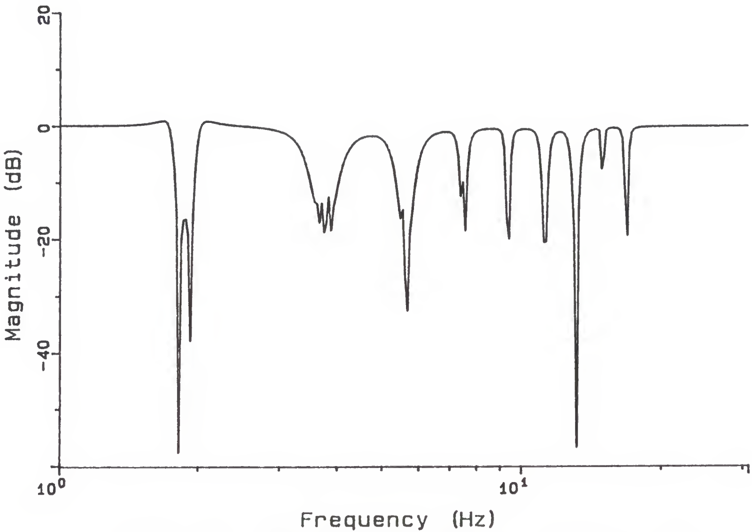


Figure 3.11. Magnitude Response of Digital Filter. Attenuation vs. frequency achieved by filters.

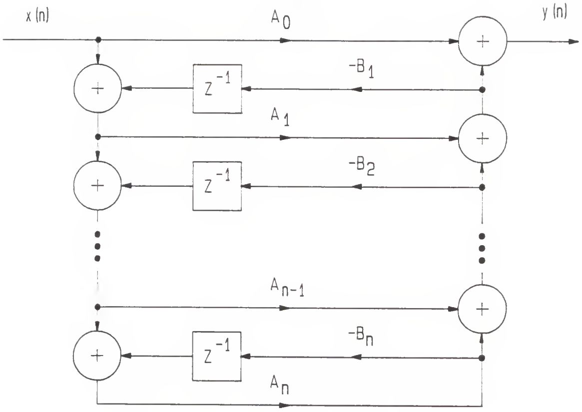


Figure 3.12. Ladder Network Used for Filter Implementation. A flow chart of discrete-time filter that's implemented in software.

The filtered data windows, shown in Figs. 3.13 - 3.15, demonstrate the influence of the signal component associated with exercise dynamics. The corresponding PSDs, shown in Figs. 3.16 - 3.18, allowed evaluation of the filter.

3.4.6 Heart Rate Analysis

The heart rate was used as an indicator of the change in physiological pressures which were assumed constant. A graph of heart rate vs. time is shown in Fig. 3.19. Heart rate was calculated from the ECG pen recording. Each value of heart rate is an average over a 5 second time interval.

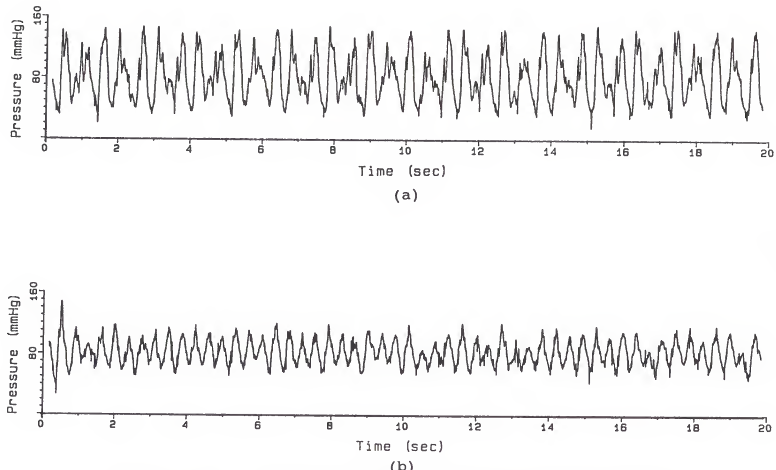


Figure 3.13. Exercise Pulmonary Arterial Pressure Data (a) Non-Filtered (b) Filtered. Demonstrates the influence of the signal component associated with exercise dynamics on the pulmonary arterial pressure.

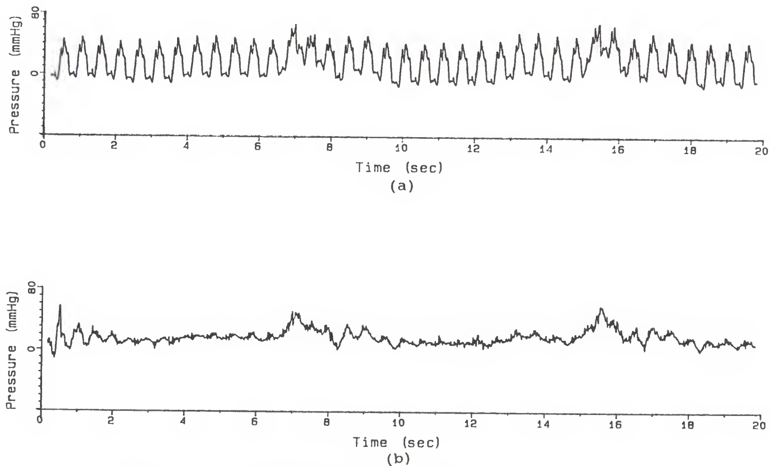


Figure 3.14. Exercise Esophageal Pressure Data (a) Non-Filtered (b) Filtered. Demonstrates the influence of the signal component associated with exercise dynamics on the esophageal pressure.

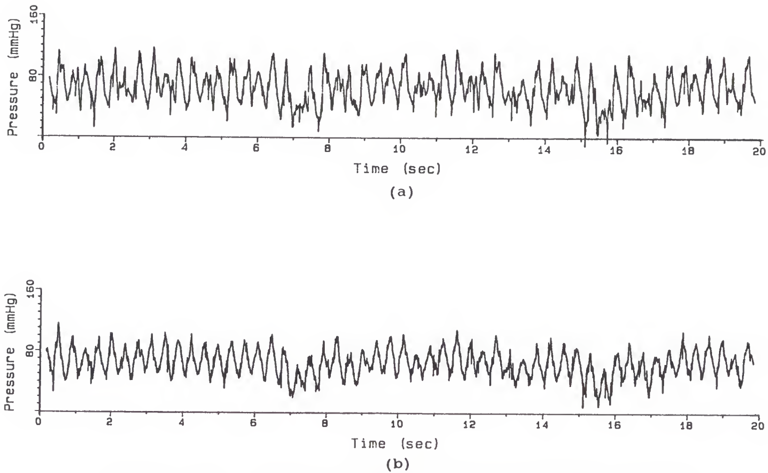


Figure 3.15. Exercise Transmural Pulmonary Artery Pressure Data (a) Non-Filtered (b) Filtered. Demonstrates the influence of the signal component associated with exercise dynamics on the transmural pulmonary artery pressure.

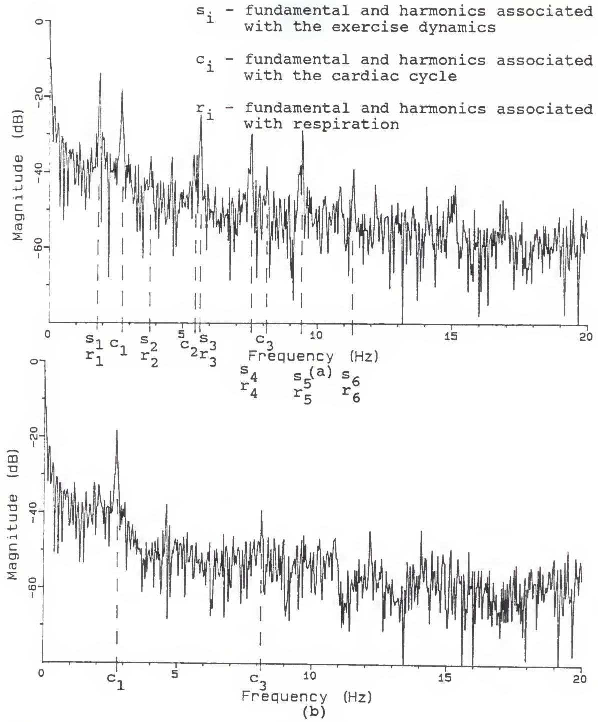


Figure 3.16. Exercise Pulmonary Arterial Pressure Power Spectral Density (a) Non-Filtered (b) Filtered. Demonstrates the filter's effect on pulmonary arterial pressure components.

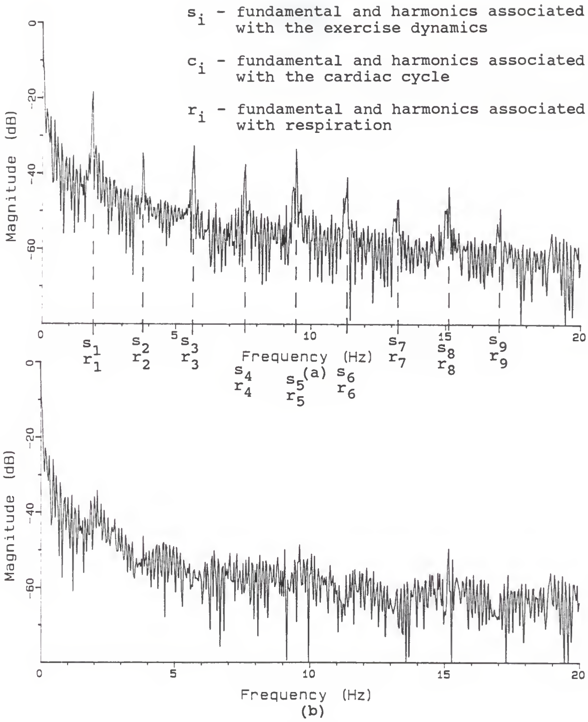


Figure 3.17. Exercise Esophageal Pressure Power Spectral Density (a) Non-Filtered (b) Filtered. Demonstrates the filter's effect on esophageal pressure components.

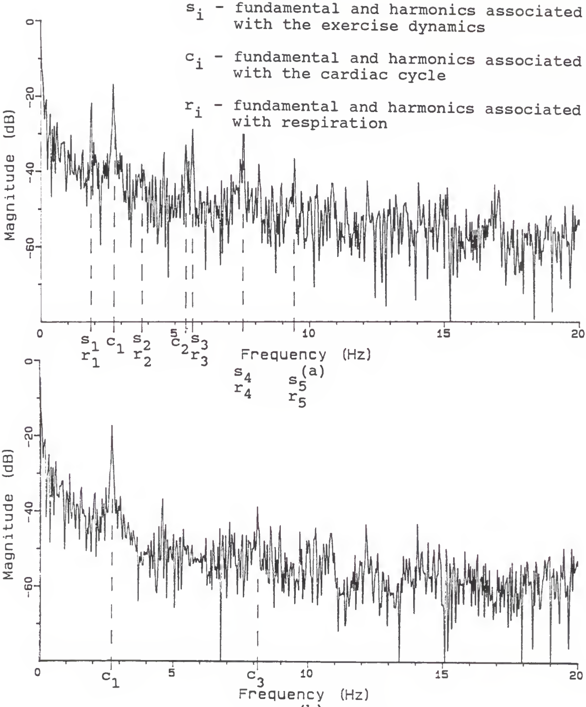


Figure 3.18. Exercise Transmural Pulmonary Artery Pressure Power Spectral Density (a) Non-Filtered (b) Filtered. Demonstrates the filter's effect on esophageal pressure components.

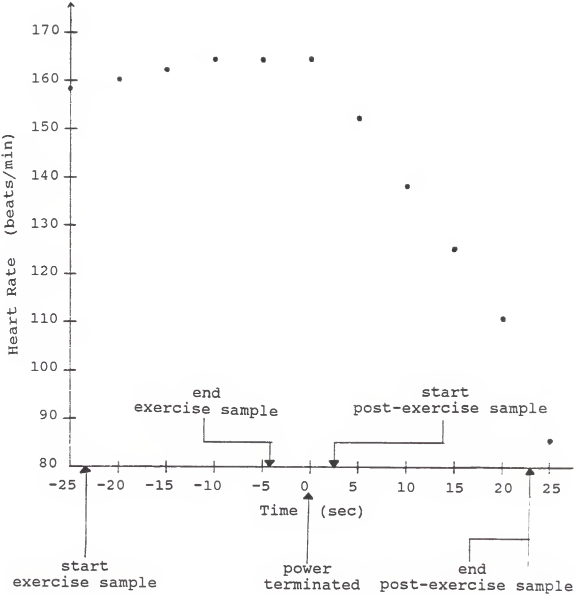


Figure 3.19. Graph of Heart Rate Calculated from Experimental Data. Illustrates the amcunt the relatively constant heart rate during exercise and the diminishing heart rate during post-exercise.

3.5 Discussion

Interpretation of the analysis results is discussed below.

3.5.1 Assumption Validity

It was assumed that physiological pressures associated with the cardiac cycle and respiration would not change significantly during the short sampling period. Fig. 3.19 shows that the heart rate was relatively constant during exercise, but diminished rapidly during the post-exercise period. For the window length chosen, 20 seconds, the heart rate dropped by 52 beats/min during the post-exercise period. The physiological pressures experienced less change over shorter window lengths; however, the accuracy of the PSD estimate decreased. Although the changing heart and respiratory rates affected the experimental results, conclusions were still possible. The diminishing heart rate resulted in the signal power of the post-exercise cardiac signal component being distributed over a range of frequencies. The maximum width of the frequency distribution range was 1/52 beats/min (0.87 Hz). Filtered results were not affected by the decreasing heart rate since only the exercise windows were filtered and the heart rate was relatively constant during exercise.

3.5.2 Power Spectral Density Interpretation

The Power Spectral Density (PSD) plot results from a method of describing a signal using frequency-domain ideas [6, 8]. A PSD shows the manner in which the average signal power is distributed over a frequency range. The PSD evaluated at a specific frequency represents the average signal power at that frequency. Thus, the PSD shows what frequency components are present in a signal. In addition, the PSD gives an estimate of the degree of influence the signal component has on the total signal. A high peak in the PSD indicates that a signal component exists at that frequency. The average power is proportional to the area under the PSD curve; thus, more power is present in a signal component spread over a large frequency range than in a signal component of equal PSD magnitude spread over a narrow frequency range.

3.5.3 Pulmonary Arterial and Esophageal Pressures

The signal components of pulmonary arterial pressure and esophageal pressure were identified with the PSDs (Figs. 3.8 and 3.9). These signal components and comparisons of those present in exercise with post-exercise signals are discussed below.

3.5.3.1 Exercise Signal Components

The PSDs of the exercise pulmonary arterial pressure (Fig. 3.8a) contained a 2.75 Hz fundamental and its 2nd and 3rd harmonics, signal component c, which corresponded to the

exercise heart rate, 165 beats/min. In each of the exercise PSDs (Figs. 3.8a and 3.9a) a 1.88 Hz fundamental and several harmonics were present. Since esophageal pressure reflects respiration, the presence of this signal component in the exercise esophageal pressure data and the interlocking of respiratory and stride frequency during running led to the conclusion that the stride/respiratory frequency was 1.88 Hz. Thus, a signal component associated with respiration, r, and a signal component associated with stride, s, were present at the 1.88 fundamental and its harmonics. The cardiac cycle, respiration, and stride did not affect the pulmonary arterial and esophageal pressures equally. This information is summarized in the following equations:

$$PA = c_{PA(2.75)} + r_{PA(1.88)} + s_{PA(1.88)} \tag{1}$$

$$ES = r_{ES(1.88)} + s_{ES(1.88)} \tag{2}$$

where PA is the pulmonary arterial pressure, and ES is the esophageal pressure.

3.5.3.2 Post-Exercise Signal Components

The average post-exercise heart rate was 118 beats/min (1.97 Hz). The cardiac component, c, was present between 1.6 - 2.5 Hz in the PSDs of the post-exercise pulmonary arterial pressure (Fig. 3.8b). The average post-exercise respiratory rate was 72 breaths/min (1.2 Hz). This fundamental and its 2nd and 3rd harmonics , r, were contained in the post-exercise esophageal pressure PSD (Fig. 3.9b). The 1.2 Hz fundamental was also present in the post-

exercise pulmonary arterial pressure PSD (3.7b). This information is summarized in the following equations:

$$PA = c_{PA(1.6-2.5)} + r_{PA(1.2)} \tag{3}$$

$$ES = r_{ES(1.2)} \tag{4}$$

3.5.3.3 Exercise/Post-Exercise Comparisons

A comparison of the exercise and post-exercise PSDs (Figs. 3.8 and 3.9) showed that the percentage of total signal power at the exercise stride/respiratory fundamental (1.88 Hz) and its harmonics was greater than the percentage of total signal power at the post-exercise respiratory fundamental (1.2 Hz) and its harmonics. This was shown by the pulmonary arterial pressure PSD (Fig. 3.8), since a larger percentage of total signal power is present at the exercise stride/respiratory fundamental (1.88 Hz) than at the exercise cardiac fundamental (2.75 Hz.) while during the post-exercise window a smaller percentage of total signal power was present at the post-exercise respiratory fundamental (1.2 Hz) than at the post-exercise cardiac fundamental (1.6 - 2.5 Hz). The esophageal pressure PSD also illustrates the larger stride/respiratory component during exercise since several harmonics are present in the exercise window, but the harmonics have negligible signal power in the post-exercise window. Since physiological pressures were assumed constant, the additional signal power in the exercise signals was due to non-physiological sources. Thus, it was concluded that the Millar transducer

recorded a signal component, s, due to the dynamics of exercise at the stride/respiratory frequency. The small percentage of signal power at the respiratory frequency in the post-exercise window (Fig. 3.8b) indicated that much of the exercise signal component with the 1.88 Hz fundamental was due to exercise dynamics.

The significance of the signal component associated with exercise dynamics, s, was shown with the pulmonary artery pressure PSD (Fig. 3.8) by comparing the signal power of the stride and respiratory components (1.88 Hz), s_{PA} and r_{PA}, to the signal power of the cardiac component (2.75 Hz), c_{PA}. A larger percentage of the total signal power was located at the 1.88 Hz fundamental than at the 2.75 Hz fundamental. In addition the harmonics of the stride/respiratory component contained considerably more power than the harmonics of the cardiac component.

Filtering the 1.88 Hz fundamental and its harmonics demonstrated the significance of the signal component associated with exercise dynamics in the time-domain. The filtered results resemble the stationary data; the differences are attributed to the changing physiological factors and removal of the signal component due to respiration. Filtering the respiration component was unavoidable since the stride frequency is equivalent to the respiratory frequency.

3.5.4 Transmural Pulmonary Artery Pressure

Under the assumption that the pulmonary artery and esophagus experienced identical pressure changes due to respiration then the transmural pulmonary artery pressure component with 1.88 Hz fundamental (Fig. 3.10a) was due entirely to non-physiological sources. This follows from the fact that transmural pulmonary artery pressure was simply the pulmonary artery pressure minus the esophageal pressure. Transmural pulmonary artery pressure calculations resulted in pressure components of equal magnitudes in pulmonary artery and esophageal pressures being canceled, i.e., the component associated with respiration. This is summarized for exercise conditions in the following equations:

$$\text{assumption:} \quad r_{PA} = r_{ES} \tag{5}$$

$$PAT = PA - ES$$

from Eq. 1 and Eq. 2

$$PAT = c_{PA(2.75)} + r_{PA(1.88)} + s_{PA(1.88)} - r_{ES(1.88)} - s_{ES(1.88)} \tag{6}$$

$$PAT = c_{PA(2.75)} + s_{PA(1.88)} - s_{ES(1.88)} \tag{7}$$

where PAT is the transmural pulmonary artery pressure. Thus, the signal differences between the filtered and non-filtered exercise transmural pulmonary artery pressures (Fig. 3.15) were due entirely to the dynamics of exercise. Removal of the dynamic component from the exercise transmural pulmonary artery signal resulted in significantly

pressures. In addition, the cardiac influence was more apparent in the resulting signal (Fig. 3.15b). The post-exercise transmural pulmonary artery window did not contain a notable signal component at the post-exercise respiratory fundamental (1.2 Hz) which the initial assumption predicted.

IV. CONCLUSIONS

A procedure was developed which allowed signal processing techniques to be applied to recorded pulmonary arterial and esophageal pressure signals and to calculated transmural pulmonary artery pressure signals. These techniques included calculating power spectral densities which show the percentage of signal power present at each frequency, and filtering, which removes the signal present at a specific frequency. By comparisons of exercise and immediately post-exercise data it was shown that non-physiological pressure sources associated with the dynamics of exercise were recorded with the Millar transducers. The significance of the dynamic pressure component was shown by the PSDs and demonstrated in the time-domain by filtering the dynamic component. The results indicated that sources associated with exercise dynamics had a substantial effect on the recorded pressures. Filtering resulted in periodic waveforms with significantly reduced peak pressures. In addition, the cardiac influence was more apparent in the filtered results. Thus, it was concluded that the dynamics of exercise lead to increased transmural pulmonary artery peaks, which may contribute to pulmonary hemorrhage.

V. RESEARCH SUGGESTIONS

This research 1) demonstrated that Millar transducer pressure recordings included significant pressures due to the dynamics of exercise, and 2) developed a procedure which enabled the location and isolation of signal components at specific frequencies. The research results suggested many areas of further research, including methods for determining the non-physiological pressure sources and improvements to the analysis procedure. The following are suggestions for continued research:

1. Collect and analyze data from a horse at the trotting exercise level. This would allow separation of the signal components of stride frequency and respiratory frequency.
2. Collect and analyze data from an exercising horse which has padded covers over its hooves. This would estimate the amount of the dynamic pressure component due to hoof impact.
3. Collect and analyze data from a horse exercising on a water treadmill. This would also estimate the amount of the dynamic pressure component due to hoof impact.
4. Collect and analyze data with an accelerometer attached at various locations on the exercising horse. This would estimate the amount of the

dynamic pressure component due to acceleration forces acting on the abdominal cavity.

5. Design and implement a comb filter which filters a fundamental and all its harmonics. This would increase the analysis speed and would decrease undesired signal attenuation due to the series of notch filters.

6. Investigate the noise level of the data acquisition system. This would provide a figure of merit for the system's dynamic range.

VI. REFERENCES

[1] Antoniou, A. Digital Filters Analysis and Design; United States of America; McGraw-Hill, Inc.; 1979 113-128.

[2] Bowron, P., F.W. Stephenson. Active Filters for Communications and Instrumentation; Maidenhead, Berkshire, England; McGraw-Hill Book Company (UK) Limited; 1979 16-19.

[3] Clarke, A.F. Review of exercise induced pulmonary hemorrhage and its possible relationship with mechanical stress. Equine Vet. J. 17(3): 166-172, 1985.

[4] Creel, E.E., Measurement of breath-by-breath oxygen consumption and carbon dioxide production in exercising calves. Master's thesis, Kansas State University, 1982.

[5] Hardin, D., Ralph User's Guide. Digital Signal Processing Package. Electrical & Computer Engineering Dept., Kansas State University, 1986.

[6] Haykin, S., Communication Systems; United States of America; John Wiley & Sons; 1983 256-270.

[7] Noyes, G., Optimization of a computerized respiratory measurement system. Master's thesis, Kansas State University, 1987.

REFERENCES (cont.)

[8] Peebles, P.Z. Jr., Probability, Random Variables, and Random Signal Principles; United States of America; McGraw-Hill, Inc.,; 1980 147-156.

[9] Phillips, C.L., H.T. Nagle, Jr., Digital Control System Analysis and Design; Englewood Cliffs, N.J.; Prentice-Hall, Inc.; 1984 373-378.

[10] Riblett, L.E. Jr., A computer-based instrumentation system for measurement of breath-by-breath oxygen consumption and carbon dioxide production in exercising humans. Master's thesis, Kansas State University, 1984.

ACKNOWLEDGEMENTS

I would like to thank my Major Professor Dr. Richard R. Gallagher for his input and guidance throughout the research and the thesis preparation. In addition I would like to thank Dr. Roger Fedde and Dr. David Leith whose insights to the physiological aspects of the research were invaluable. Thanks also go to Kipp Erickson for collecting the data and for answering my numerous questions concerning physiology. I would also like to thank Dr. Ruth Dyer for her help and suggestions.

I also greatly appreciate the assistance and advice I received from Durwin Nigus while preparing this thesis.

A special thanks goes to my mother Charlene, my brother Brant, and my sisters Deloris, Denise, and Cyndi. Their support has been unfaltering throughout my education.

APPENDIX A

USER'S MANUAL

A.1 Scope

The primary purpose of this manual is to enable the user to perform specific data analysis which involve selected signal processing techniques, i.e., power spectral density estimations, digital filtering, and plotting the results. Prior exposure to the research and the VAX 11/750 system are assumed.

The analysis software operates on the VAX 11/750 system in the Department of Electrical and Computer Engineering, Kansas State University, Manhattan, KS. For a description of the routines used in the analysis software refer to the program listings in Appendix B.

A.2 Analysis Procedure

Before beginning the data analysis, the data must be collected, digitized, and stored in ASCII format on the VAX 11/750 system [7].

During data collection all data are inverted (a value x is recorded as $2^{12} - x$). Thus, to obtain the actual data values the data files are first inverted. Next, transmural pulmonary artery pressure is calculated for the non-filtered and filtered data. The data files are converted to SG format to accommodate the digital signal processing package. The data are then processed by cascaded digital filters to

attenuate selected frequency components. Plots of the non-filtered and filtered data are then generated. Plots of power spectral densities of the non-filtered and filtered data are obtained using the digital signal processing package. A flow chart of the analysis procedure is shown in Fig. A.1. The data analysis procedure is summarized below:

Step 1: Invert the data files.

Step 2: Calculate transmural pulmonary artery pressure data.

Step 3: Convert the data files to SG format.

Step 4: Process the data through a series of digital bandstop filters.

Step 5: Plot the non-filtered and filtered data.

Step 6: Plot power spectral densities using the digital signal processing package, RALPH.

A.3 Sample Run

To illustrate the analysis procedure a sample run is included and discussed. Computer prompts are indented and the user's responses are indented in bold face printing. The sample run analyzes only the pulmonary arterial pressure during exercise. Table A.1 contains a list and brief description of all the data files used in the sample run.

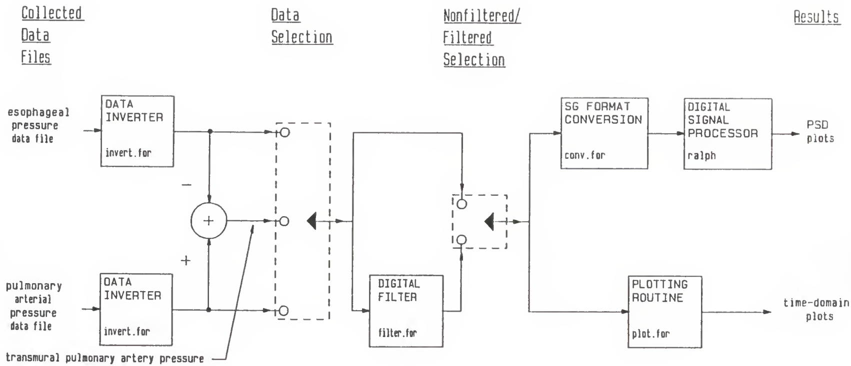


Figure A.1. Data Analysis Procedure Flow Chart. This depicts the analysis procedure illustrating the sequence of events and routines used to obtain PSD and time-domain plots.

A.3 Sample Run: Pulmonary Arterial Pressure (cont.)

Table A.1. Sample Run Data Files

Filename	Description
PULMRUN.ASC	pulmonary arterial pressure data collected during exercise
PULMRUNI.DAT	pulmonary arterial pressure during exercise, obtained by inverting the collected data
RPULMRUNI.DAT	pulmonary arterial pressure during exercise in SG format, obtained by reformatting the inverted pressure data
PULMRUNIF.DAT	filtered pulmonary arterial pressure during exercise, obtained by processing the data by a digital filter (this procedure is described in the thesis)
RPULMRUNIF.DAT	filtered pulmonary arterial pressure during exercise in SG format, obtained by reformatting the filtered pressure data
ESOPRUNI.DAT	esophageal pressure during exercise, obtained by inverting the collected data
TRANRUNI.DAT	transmural pulmonary artery pressure during exercise, obtained from the pulmonary arterial and esophageal pressures
ZZZZZZ.DAT	temporary file used to store filtered data
MAG.DAT	magnitude response of selected digital filter

A.3 Sample Run: Pulmonary Arterial Pressure (cont.)

The analysis could also be performed on pulmonary arterial post-exercise, esophageal exercise and post-exercise, and transmural pulmonary artery exercise and post-exercise pressures.

The sample run analyzes the data contained in the ASCII file PULMRUN.ASC. For brevity the sample run attenuates only two arbitrarily selected frequency components, 1.88 Hz and 9.4 Hz. Removal of additional frequencies is simply repeating the illustrated procedure. The magnitude response of the filter is plotted on the 4014 Tektronix screen. Plots of non-filtered pulmonary arterial pressure, filtered pulmonary arterial pressure and the corresponding power spectral densities are generated using the HP7475 plotter.

Step 1:

```
RUN INVERT <CR>
INPUT DATA FILENAME = ?
'PULMRUN.ASC' <CR>
OUTPUT DATA FILENAME = ?
'PULMRUNI.DAT'<CR>
```

Step 2:

```
RUN TRAN <CR>
PULMONARY FILENAME = ?
'PULMRUNI.DAT' <CR>
ESOPHAGEAL FILENAME = ?
'ESOPRUNI.DAT' <CR>

TO OBTAIN THE TRANSMURAL PULMONARY ARTERY
PRESSURE IN mmHg, THE DATA WILL BE SCALED USING

Y=A1*PULM. PRESSURE+B1-A2*ESOP. PRESSURE-B2
```

```
WHERE A1 &B1 ARE SCALING FACTORS USED TO CONVERT
PULMONARY ARTERIAL PRESSURE DATA TO mmHg

WHERE A2 & B2 ARE SCALING FACTORS USED TO CONVERT
ESOPHAGEAL PRESSURE DATA TO mmHg

A1 = ?
54.9512 <CR>
B1 = ?
0.0156 <CR>
A2 = ?
0.0353 <CR>
B2 = ?
-65.3908 <CR>
TRANSMURAL FILENAME = ?
'TRANRUNI.DAT' <CR>
```

Transmural pulmonary artery pressure in mmHg is obtained from the pulmonary arterial and esophageal pressures. To obtain the pressure values in mmHg it is necessary to scale the digitized data. The values A1 and B1 are the values necessary to scale the pulmonary arterial pressure data to pressure in mmHg. These values are calculated from the experimental calibration signals and the corresponding digital values. Similarly the values of A2 and B2 are calculated from the esophageal pressure information. Transmural pulmonary artery pressure calculations require an esophageal pressure data file. This sample run assumes that the file ESOPRUNI.DAT contains esophageal pressure during exercise.

Step 3:

```
RUN CONV <CR>
INPUT DATA FILENAME = ?
```

```
'PULMRUNI.DAT' <CR>
OUTPUT FILENAME FOR SAMPLED DATA = ?
'RPULMRUNI.DAT' <CR>
```

This step is necessary because the digital signal processing package which is used requires all input files be in SG format.

Step 4:

```
RUN INIT <CR>
```

This step clears the data file used for the magnitude response. To prevent any previous data from causing erroneous results this file must be cleared prior to filtering.

```
RUN FILTER <CR>
DENOMINATOR DEGREE = ?
2 <CR>
A01 = ?
7.464 <CR>
B01 = ?
0.9989 <CR>
B11 = ?
1.1701 <CR>
```

The routine requires the user to enter the coefficients of a 2 or 4 pole elliptic filter. Factors effecting filter selection include ripple, gain, and stability. Elliptic filter coefficient values are available from numerous sources [2].

```
CENTER FREQUENCY (HZ) = ?
1.88 <CR>
BANDWIDTH (HZ) = ?
0.30 <CR>
```

The bandwidth selected is the minimum which allows filter stability. This value will depend on the center frequency and the elliptical filter coefficients.

```
BEGINNING DECADE = ?
1 <CR>
INCREMENTS PER DECADE = ?
150 <CR>
```

These values determine the frequency range of the magnitude response plot. All frequencies which are to be filtered should be included in the range. The routine plots 300 points, thus the frequency range for these values is 1 - 100 Hz.

```
ENTER A 1 TO BEGIN THE MAGNITUDE RESPONSE PLOT
1 <CR>
DEVICE = ? 4014 OR 7475
4014 <CR>
ENTER A 1 TO BEGIN FILTERING THE DATA
1 <CR>
INPUT DATA FILENAME = ?
'PULMRUNI.DAT' <CR>
OUTPUT FILENAME FOR SAMPLED DATA
'RPULMRUNIF.DAT' <CR>

RUN FILTER <CR>
```

The filtering procedure is repeated to remove the 9.4 Hz frequency component.

```
DENOMINATOR DEGREE = ?
4 <CR>
A01 = ?
10193.2448371366 <CR>
B01 = ?
6.53565094649181OE-3 <CR>
B11 = ?
0.160452726787879 <CR>
```

A.3 Sample Run: Pulmonary Arterial Pressure (cont.)

```
A02 = ?
100280.396030769 <CR>
B02 = ?
6.44752251188016E-3 <CR>
B12 = ?
0.160467075995419 <CR>
```

Filter selection is based on the criteria discussed earlier. These filter coefficients are generated with the routine ELLIPTIC which implements an elliptic approximation [1].

```
CENTER FREQUENCY (HZ) = ?
9.4 <CR>
BANDWIDTH (HZ) = ?
0.011 <CR>
```

These values are selected using the criteria discussed earlier.

```
BEGINNING DECADE = ?
1 <CR>
INCREMENTS PER DECADE = ?
150 <CR>
```

These values must be the same during the removal of each frequency component. This is necessary since the filter responses are shown on one plot.

```
ENTER A 1 TO BEGIN MAGNITUDE RESPONSE PLOT
1 <CR>
DEVICE = ? 4014 OR 7475
4014 <CR>
ENTER A 1 TO BEGIN FILTERING THE DATA
1 <CR>

ENTER DATA FILENAME
'ZZZZZZ.DAT' <CR>
OUTPUT FILENAME FOR SAMPLED DATA
'RPULMRUNIF.DAT' <CR>
```

Filtered data are stored in a temporary file named ZZZZZZ.DAT and also in SG format with a user selected filename. Since several digital filters are being used, the input of the second filter is the output of the first filter. Thus, the data inputted to the second filter is stored in the temporary file ZZZZZZ.DAT.

```
RENAME <CR>
FROM:
'ZZZZZZ.DAT' <CR>
TO:
'PULMRUNIF.DAT' <CR>
```

After completing the filtering process, the temporary file should be renamed to prevent further analysis from destroying the filtered data.

Step 5:

```
RUN PLOT <CR>
LOWER PLOT TITLE = ?
'PULMONARY EXERCISE DATA' <CR>
LOWER DATA FILENAME = ?
'PULMRUNI.DAT' <CR>
```

Two plots will be generated using either the 4014 Tektronix screen or the HP7475 plotter. If the plotter is selected an 11 by 17 inch paper (size B) must be used.

```
THE PLOTS WILL BE SCALED USING Y = A + B*X
A = ?
54.9512 <CR>
B = ?
0.0156 <CR>
```

A.3 Sample Run: Pulmonary Arterial Pressure (cont.)

Scaling is used to convert the digitized data to the pressure value in mmHg. Note that the transmural pulmonary artery pressure is saved in mmHg of pressure, thus should not be scaled. The scaling values are determined from the experimental calibration signals and their corresponding digitized values.

```
DEVICE = ? 4014 OR 7475
7475 <CR>
FIRST Y-AXIS VALUE = ?
0 <CR>
LAST Y-AXIS VALUE = ?
200 <CR>
```

The y-axis values approximate the range of the pressure signal in mmHg.

```
UPPER DATA FILENAME = ?
'PULMRUNIF.DAT' <CR>
UPPER PLOT TITLE = ?
'FILTERED PULMONARY EXERCISE DATA' <CR>
```

Step 6:

RALPH is used to obtain power spectral density plots of the non-filtered and filtered data. The corresponding input data files are RPULMRUNI.DAT and RPULMRUNIF.DAT. For information on using RALPH, refer to RALPH User's Manual [5].

APPENDIX B

PROGRAM LISTINGS

This appendix contains the listings for the routines used for this research. All routines are in Fortran and were run on the VAX 11/750 EECE Dept. Kansas State University, Manhattan, KS. The routines included are:

```
************************************************************************
*
*   ROUTINE:       Mainline
*                  CONV
*
*   DESCRIPTION:   This routine retrieves data from disk, converts
*                  it to SG format, and saves the results on disk.
*
*   ARGUMENTS:     None
*
*   ROUTINES
*   CALLED:        None
*
*   AUTHOR:        Deanna L. Carroll
*                  Rt 1
*                  Lewis, KS 67552
*                  (316) 324-5338
*
*   DATE CREATED:  10Aug88
*
************************************************************************
      IMPLICIT NONE
      REAL XS(6000)
      INTEGER X,XP(6000)
      CHARACTER*20 FILENAME

*  Retrieve the input data from disk

      PRINT*, 'INPUT DATA FILENAME =  ?'
      READ*, FILENAME
      OPEN (UNIT=1, STATUS='OLD', FILE=FILENAME, RECORDTYPE=
     + 'VARIABLE', CARRIAGECONTROL='NONE', ACCESS=
     + 'SEQUENTIAL')
      READ (UNIT=1,FMT=1000) XP
      CLOSE (UNIT=1, STATUS='KEEP')

*  Change integer data to real data

      DO X=1,6000,1
         XS(X)=REAL(XP(X))
      END DO

*  Convert data to SG format and save on disk

      CALL SGOPEN (1, 'WRITE', '>>>OUTPUT FILENAME FOR SAMPLED
     +    DATA=','NONAME', 'REAL', 6000)
      CALL SGTRAN (1, 'WRITE', 'REAL', XS, 6000)

      STOP
1000  FORMAT(I12)
      END
```

```
************************************************************************
*
*   ROUTINE:        Mainline
*                   ELLIPTIC
*
*   DESCRIPTION:    This routine generates an elliptic normalized
*                   lowpass transfer function.  The elliptic
*                   approximation used was obtained from "Digital
*                   Filters Analysis and Design" by Andreas
*                   Antoniou.  This routine requires the user to
*                   enter values of selectivity factor and minimum
*                   stopband loss which can be obtained from the
*                   above reference.  Only transfer functions with
*                   an even number of poles can be generated.
*
*   ARGUMENTS:      None
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:  29July88
*
************************************************************************

      implicit none
      integer n,r,i,m
      real*8 ap,aa,k,k1,q0,q,gamma,sigma,num,denom,w,ohm,mu,
     +   v,a0(4),b0(4),b1(4),h0

*  ENTER ELLIPTIC TRANSFER FUNCTION INFORMATION

      print*,'# of poles  = ?'
      read*, n
      print*,'selectivity factor, k, = ?'
      read*, k
      print*,'minimum stopband loss, aa, = ?'
      read*, aa

      ap=0.5
      r=n/2
      k1=dsqrt(1.0-k**2.0)
      q0=0.5*(1.0-dsqrt(k1))/(1.0+dsqrt(k1))
      q=q0+2*q0**5.0+15.0*q0**9.0+150.0*q0**13.0
      gamma=(1.0/2.0*n)*dlog((10.0**(0.05*ap)+1)/(10.0**
     +    (0.05*ap)-1))
      num=0.0
      denom=0.0
      do m=0,5,1
          num=2.0*q**0.25*((-1.0)**m)*q**(m*(m+1))*dsinh((2.0*
     +       m+1.0)*gamma)+num
          denom=2.0*((-1.0)**m)*q**(m**2)*dcosh(2.0*m*gamma)+denom
      end do
      denom=denom+1
      sigma=dabs(num/denom)
      w=sqrt((1.0+k*sigma**2)*(1.0+(sigma**2/k)))
```

```
*  CALCULATE TRANSFER FUNCTION COEFFICIENTS

      do i=1,r,1
          mu=i-0.5
          num=0.0
          denom=0.0
          do m=1,5,1
              num=2*q**.25*((-1)**m)*q**(m*(m+1))*dsin((2*m+1)*
     +            3.14159*mu/n)+num
              denom=2*((-1)**m)*q**(m**2)*dcos(2*m*3.14159*mu/n)+denom
          end do
          ohm=num/(1+denom)
          v=dsqrt((1-k*ohm**2)*(1-ohm**2/k))
          a0(i)=1.0/ohm**2
          b0(i)=((sigma*v)**2+(ohm*w)**2)/((1+sigma**2*ohm**2)**2)
          b1(i)=2*sigma*v/(1+sigma**2*ohm**2)
      end do

*  MULTIPLIER CONSTANT

          h0=10.0**((-.05)*ap)
      do i=1,r,1
          h0=h0*b0(i)/a0(i)
      end do

*  DISPLAY TRANSFER FUNCTION COEFFICIENTS

      print *,'h0=',h0
      do i=1,r,1
          print*, 'i=',i
          print*, 'a0=',a0(i)
          print*, 'b0=',b0(i)
          print*, 'b1=',b1(i)
      end do

      stop
      end
```

```
************************************************************************
*
*   ROUTINE:       Mainline
*                  FILTER
*
*   DESCRIPTION:   This program finds the discrete time transfer
*                  function of a 4 or 8 pole bandstop filter,
*                  plots the filter's dt magnitude response, and
*                  implements the digital filter.  The user is
*                  required to enter a 2 or 4 pole elliptic
*                  transfer function, center frequency,
*                  bandwidth, plot variables, and the filename of
*                  the input data.  The magnitude response is
*                  composed of the response of the entered filter
*                  and previous response information stored in
*                  the file "MAG.DAT".  The magnitude response is
*                  saved in "MAG.DAT".
*
*   ARGUMENTS:     None
*
*   ROUTINES
*   CALLED:        SUBROUTINES
*
*                  dt_response(degnum,num,degdenom,denom,numfreq,
*                      frequencies,response)
*
*                  simple_plot(numfreq,freqsc,dbmagsc,xtitle,xunits,
*                      ytitle,yunits,title,plot_type)
*
*                  digital(deg,denom,num,filename,numdata)
*
*   AUTHOR:        Deanna L. Carroll
*                  Rt 1
*                  Lewis, KS 67552
*                  (316) 324-5338
*
*   DATE CREATED:  14July88
*
************************************************************************

      implicit none
      integer bico,blt,bltdeg,bsdegdenom,bsdegnum,error,i,j,k,
     +     highpass_to_bandstop,lpdegdenom,hpdegdenom,hpdegnum,
     +     numfreq,numdata
      real*8 bltdenom(0:30),bltnum(0:30),bs_bandwidth,bsdenom(0:30),
     +     bs_f_center,bsnum(0:30),lpdenom(0:30),lpnum(0:30),
     +     hpdenom(0:30),hpnum(0:30),skewwa1,skewwa2,factln,gammln,
     +     flow,fup,cmag,cmagsq,dbmag(300),frequencies(300),a1,a2,
     +     b11,b10,b21,b20,var1,var2,mag(300)
      real dbmagsc(300),freqsc(300)
      complex*16 response(300)
      character*16 filename
      character*40 title,xtitle,xunits,ytitle,yunits,plot_type

*  SET DESIRED ELLIPTIC TRANSFER FUNCTION

      print*, 'denominator degree = ?'
      read*,lpdegdenom
      print*,'a01 = ?'
      read*,a1
      a1=10193.2448371366
      print*,'b01 = ?'
      read*,b10
      b10=6.535650946619810e-3
```

```
      print*,'b11 = ?'
      read*,b11
      b11=0.160452726787879
      if (lpdegdenom.ne.2) then
          print*,'a02 = ?'
          read*,a2
          a2=100280.396030769
          print*,'b02 = ?'
          read*,b20
          b20=6.447522511880168e-3
          print*,'b12 = ?'
          read*,b21
          b21=0.160467075995419
      else
      end if

*  CALCULATE LOW-PASS FILTER COEFFICIENTS

      if (lpdegdenom.eq.2) then
          lpnum(2)=1.0*b10
          lpnum(1)=0.0
          lpnum(0)=a1*b10
          lpdenom(2)=1.*a1
          lpdenom(1)=b11*a1
          lpdenom(0)=b10*a1
      else
          lpnum(4)=b20*b10
          lpnum(3)=0.0
          lpnum(2)=b20*b10*(a1+a2)
          lpnum(1)=0.0
          lpnum(0)=b20*b10*a1*a2
          lpdenom(4)=a2*a1
          lpdenom(3)=a2*a1*(b21*b11)
          lpdenom(2)=a2*a1*(b20+b11*b21+b10)
          lpdenom(1)=a2*a1*(b11*b20+b10*b21)
          lpdenom(0)=a2*a1*b10*b20
      end if

*  SET FILTER PARAMETERS

      print*,'center frequency (Hz) = ?'
      read*,var1
      bs_f_center=2*3.14159*var1
      print*,'bandwidth (Hz) = ?'
      read*,var2
      bs_bandwidth=2*3.14159*var2
      flow=var1-var2/2.0
      fup=var1+var2/2.0
      numdata=6000

*  TRANSFORM LOW-PASS FILTER TO HIGH-PASS FILTER

      hpdegnum=lpdegdenom
      hpdegdenom=lpdegdenom
      do i=0,lpdegdenom,1
          hpnum(lpdegdenom-i)=lpnum(i)
          hpdenom(lpdegdenom-i)=lpdenom(i)
      end do
```

```
*  INCLUDE SKEW FACTOR WHICH IS NECESSARY FOR DISCRETE TIME

      skewwa1=tan(flow*3.14159/300.0)
      skewwa2=tan(fup*3.14159/300.0)
      bs_bandwidth=skewwa2-skewwa1
      bs_f_center=dsqrt(skewwa1*skewwa2)

*  TRANSFORM HIGH-PASS FILTER TO BAND-STOP FILTER

      error=highpass_to_bandstop(hpdegnum,hpnum,hpdegdenom,
     +     hpdenom,bs_f_center,bs_bandwidth,bsdegnum,bsnum,
     +     bsdegdenom,bsdenom)

*  PERFORM BILINEAR TRANSFORMATION

      error=blt(bsdegnum,bsnum,bsdegdenom,bsdenom,bltdeg,
     +     bltnum,bltdenom)


*  SET UP PLOT FREQUENCIES FOR MAGNITUDE RESPONSE

      numfreq=300
      print*,'beginning decade = ?'
      read*, var1
      print*,'increments per decade = ?'
      read*,var2
      k=1
      do while (k.le.numfreq)
          j=0
          do while ((j.le.var2).and.(k.le.numfreq))
                  frequencies(k)=var1*10**(j/var2)
                  freqsc(k)=sngl(frequencies(k))
                  k=k+1
                  j=j+1
          end do
          var1=var1*10
      end do

*  EVALUATE DISCRETE TIME RESPONSE

      call dt_response(bltdeg,bltnum,bltdeg,bltdenom,numfreq,
     +     frequencies,response)

*  SET PLOT VALUES FOR MAGNITUDE RESPONSE

     plot_type='linear-log'
     ytitle='Magnitude'
     yunits='dB'
     title=' '
     xtitle='Frequency'
     xunits='Hz'

*  CALCULATE dB MAGNITUDE OF COMBINED FILTERS

*    RETRIEVE PREVIOUS FILTER RESPONSE
      open(unit=1,status='old',file='mag.dat',recordtype=
     + 'variable',carriagecontrol='none',access='sequential')
      read(unit=1,fmt=1000) mag
      close(unit=1,status='keep')

*    CALCULATE FILTER RESPONSE
      do i=1,numfreq,1
          dbmag(i)=20*log10(cmag(response(i)))
          dbmagsc(i)=sngl(dbmag(i))+mag(i)
```

```
         mag(i)=dbmagsc(i)
      end do

*    SAVE FILTER RESPONSE
      open(unit=1,status='old',file='mag.dat',recordtype=
     +  'variable',carriagecontrol='none',access='sequential')
      write(1,fmt=1000) mag
      close(unit=1,status='keep')

*   OPTION TO BEGIN DT MAGNITUDE RESPONSE PLOT

      print*, 'enter a 1 to begin magnitude response plot'
      read*,i
      if (i.eq.1) then
         call simple_plot(numfreq,freqsc,dbmagsc,xtitle,
     +         xunits,ytitle,yunits,title,plot_type)
      end if

*   OPTION TO OBTAIN FILTERED OUTPUT

      print*, 'enter a 1 to begin filtering the data '
      read*,i
      if (i.eq.1) then
         print*, 'input data filename = ?'
         read*, filename
         call digital(bltdeg,bltdenom,bltnum,filename,numdata)
      end if

      stop
1000  format(f8.3)
      end
```

```
************************************************************************
*
*   ROUTINE:        Function
*                   BICO(n,k)
*
*   DESCRIPTION:    This function returns the binomial coefficient
*                   (k of n) as a floating-point number.
*
*   ARGUMENTS:
*       k           (input)  integer
*
*       n           (input)  integer
*
*   ROUTINES
*   CALLED:         FUNCTIONS
*
*                   factln(n)
*
*   AUTHOR:         Numerical Recipes The Art of Scientific
*                   Computing Cambridge University Press
*
************************************************************************

      integer function bico(n,k)

      implicit none
      integer n,k
      real*8 factln,gammln

      bico=anint(exp(factln(n)-factln(k)-factln(n-k)))

      return
      end
```

```
**************************************************************************
*
*   ROUTINE:        Function
*                   BLT(degnum,num,degdenom,denom,bltdeg,bltnum,
*                       bltdenom)
*
*   DESCRIPTION:    An integer function which performs a bilinear
*                   transformation on the transfer function H(s) of
*                   the linear system.
*
*   ARGUMENTS:
*       bltdeg      (output)  integer
*                   The degree of the numerator and the denominator
*                   polynomials in the resulting transfer function H(z).
*
*       bltdenom    (output)  real
*                   An array containing the denominator coefficients
*                   of H(z).
*
*       bltnum      (output)  real
*                   An array containing the numerator coefficients
*                   of H(z).
*
*       degdenom    (input)  integer
*                   The degree of the denominator polynomial of the
*                   system transfer function H(s).
*
*       degnum      (input)  integer
*                   The degree of the numerator polynomial of the
*                   continuous transfer function H(s).
*
*       denom       (input)  real
*                   An array containing the coefficients of the
*                   denominator polynomial of H(s).
*
*       num         (input)  real
*                   An array containing the coefficients of the
*                   numerator polynomial of H(s)
*
*   ROUTINES
*   CALLED:         function bico
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:   14July88
*
**************************************************************************


      integer function blt(degnum,num,degdenom,denom,bltdeg,
     +    bltnum,bltdenom)
      implicit none
      integer alpha,bico,bltdeg,c(0:30,0:30),degdenom,degnum,error,
     +    k,m,n
      real*8 bltdenom(0:30),bltnum(0:30),denom(0:30),factln,
     +    gammln,num(0:30)

* THIS ROUTINE WILL HANDLE POLYNOMIALS WITH A MAX DEGREE OF 30

      alpha=max(degnum,degdenom)
```

```
*  COMPUTE THE SET OF COEFFICIENTS Ck,n

      do k=0,alpha,1
          do n=0,alpha,1
              if (k.eq.0) then
                  c(k,n)=1
              else
                  if (n.eq.0) then
                      c(k,n)=bico(alpha,k)
                  else
                      c(k,n)=-c(k-1,n)+c(k,n-1)-c(k-1,n-1)
                  end if
              end if
          end do
      end do

*  COMPUTE THE NUMERATOR COEFFICIENTS

      do k=0,alpha,1
          do n=0,degnum,1
              bltnum(k)=num(n)*c(k,n)+bltnum(k)
          end do
      end do

*  COMPUTE THE DENOMINATOR COEFFICIENTS

      do k=0,alpha,1
          do n=0,degdenom,1
              bltdenom(k)=denom(n)*c(k,n)+bltdenom(k)
          end do
      end do

      bltdeg=alpha
      open(unit=1,file='coeff.dat',status='unknown')
      do k=0,bltdeg,1
          write (1,*) bltnum(k),bltdenom(k)
      end do
      close (unit=1,status='keep')

      error=0
      return
      end
```

```
***************************************************************************
*
*   ROUTINE:        Function
*                   CMAG(x)
*
*   DESCRIPTION:    A real funciton which returns the magnitude of
*                   its complex-valued argument x.
*
*   ARGUMENTS:
*       x           (input)  complex
*                   The complex numger whose magnitude is to be
*                   evaluated.
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:   15July88
*
***************************************************************************


      real*8 function cmag(x)
      implicit none
      complex*16 x

      cmag=cdabs(x)

      return
      end
```

```
*************************************************************************
*
*   ROUTINE:        Subroutine
*                   DIGITAL(deg,denom,num,filename,numdata)
*
*   DESCRIPTION:    This routine implements a digital filter using
*                   a ladder network.  The filter degree must not
*                   exceed 30.  The filter's output is saved in the
*                   datafile "zzzzzz.dat" and using SG format in
*                   a user's selected datafile.
*
*   ARGUMENTS:
*       deg         (input)  integer
*                   The degree of the filter.
*
*       denom       (input)  real
*                   An array containing the denominator
*                   coefficients of the filter.
*
*       filename    (input)  character*16
*                   The filename where the data to be processed is
*                   saved.
*
*       num         (input)  real
*                   An array containing the numerator coefficients
*                   of the filter.
*
*       numdata     (input)  integer
*                   The number of data points to be processed.
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:  20July88
*
*************************************************************************

      subroutine digital(deg,denom,num,filename,numdata)

      implicit none
      integer i,j,deg,cnt,flag,numdata,xp(6000),yf(6000)
      real*8 num(0:30),denom(0:30),temp(0:30),a(0:30),b(0:30),x(6000),
     +     s(59),norm
      real y(6000)
      character*16 filename

      PRINT*,DEG
* NORMALIZE TRANSFER FUNCTION

      norm=denom(0)
      do i=0,deg,1
          num(i)=num(i)/norm
          denom(i)=denom(i)/norm
      end do
```

```
*  READ DATA SEQUENCE

      open (unit=1,status='old',file=filename,recordtype=
     +     'variable',carriagecontrol='none',access='sequential')
      read (unit=1,fmt=1000) xp
      close (unit=1,status='keep')
      do i=0,numdata-1,1
          x(i)=dble(real(xp(i)))
      end do

*  DESIGN FILTER USING LADDER NETWORK

      cnt=deg
      i=0
      do while(flag.eq.0)
      a(i)=num(cnt)/denom(cnt)
      do j=0,cnt,1
          temp(j)=denom(j)
          denom(j)=num(j)-a(i)*denom(j)
          num(j)=temp(j)
      end do
      if (i.ne.deg) then
          b(i+1)=num(cnt)/denom(cnt-1)
          do j=0,cnt,1
              temp(j)=denom(j)
          end do
          do j=cnt,1,-1
              denom(j)=num(j)-b(i+1)*denom(j-1)
          end do
          denom(0)=num(0)
          do j=0,cnt,1
              num(j)=temp(j)
          end do
          cnt=cnt-1
          i=i+1
      else
          flag=1
      end if
      end do

*  FILTER DATA

      do i=1,6000,1
          s(1)=x(i)-b(1)*s(2*deg-1)
          do j=2,deg-1,1
              s(j)=s(j-1)-b(j)*s(2*deg-j)
          end do
          s(deg)=s(deg-1)-b(deg)*a(deg)*s(deg)
          s(deg+1)=a(deg)*s(deg)+a(deg-1)*s(deg-1)
          do j=deg+2,2*deg-1,1
              s(j)=s(j-1)+a(2*deg-j)*s(2*deg-j)
          end do
          y(i)=sngl(s(2*deg-1)+a(0)*x(i))
          yf(i)=nint(y(i))
          end do

*  SAVE FILTER OUTPUT DATA

      open (unit=1,status='new',file='zzzzzz',recordtype=
     +     'variable',carriagecontrol='none',access='sequential')
      write (1,1000) yf
      close(unit=1,status='keep')
```

```
*  SAVE FILTER OUTPUT DATA IN SG FORMAT

      CALL SGOPEN (1,'WRITE','>>>OUTPUT FILENAME FOR SAMPLED DATA=',
     +     'NONAME','REAL',6000)
      CALL SGTRAN (1,'WRITE','REAL',Y,6000)

      return
1000  format(i12)
      end
```

```
************************************************************************
*
*   ROUTINE:        Subroutine
*                   dt_response(degnum,num,degdenom,denom,numfreq,
*                       frequencies,response)
*
*   DESCRIPTION:    A subroutine which evaluates the frequency
*                   response of a discrete time linear system.
*
*   ARGUMENTS:
*
*       degdenom    (input)  integer
*                   The degree of the denominator polynomial of the
*                   system transfer function H(s).
*
*       degnum      (input)  integer
*                   The degree of the numerator polynomial of the
*                   system transfer function H(s).
*
*       denom       (input)  real
*                   An array containing the coefficients of the
*                   denominator polynomial in the system transfer
*                   function H(s).
*
*       frequencies
*                   (input)  real
*                   An array containing the frequencies at which
*                   the response is to be evaluated.
*
*       numfreq     (input)  integer
*                   The number of frequencies at which the response
*                   is to be evaluated.
*
*       num         (input)  real
*                   An array containing the coefficients of the
*                   numerator polynomial of the system transfer
*                   function H(s).
*
*       response    (output)  complex
*                   An array of the values (real and imaginary parts)
*                   of the frequency response, evaluated at the
*                   frequencies inputted.
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:   16July88
*
************************************************************************


      subroutine dt_response(degnum,num,degdenom,denom,numfreq,
     +     frequencies,response)

      implicit none
      integer degdenom,degnum,i,j,numfreq
      real*8 denom(0:30),frequencies(300),num(0:30),t,w,wvalue(300)
      complex*16 denomm,numm,response(300),z
```

```
*   CONVERT FREQUENCIES TO RAD/SEC

      do i=1,numfreq,1
          wvalue(i)=2*3.1415927*frequencies(i)
      end do

*   SET THE PERIOD OF THE SAMPLE

       t=1.0/300

*   CALCULATE THE RESPONSE FOR EACH FREQUENCY

      do i=1,numfreq,1
          z=dcmplx(dcos(t*wvalue(i)),dsin(t*wvalue(i)))

*   CALCULATE THE NUMERATOR OF THE RESPONSE

          numm=num(0)
          do j=1,degnum,1
              numm=numm+num(j)*(z**(-1*j))
          end do

*   CALCULATE THE DENOMINATOR OF THE RESPONSE

          denomm=denom(0)
          do j=1,degdenom,1
              denomm=denomm+denom(j)*(z**(-1*j))
          end do

*   COMBINE NUMERATOR AND DENOMINATOR TO FORM RESPONSE

          response(i)=numm/denomm
      end do

      return
      end
```

```
*************************************************************************
*
*   ROUTINE:        Function
*                   FACTLN(n)
*
*   DESCRIPTION:    This function returns the value of ln(n!)
*
*   ARGUMENTS:
*      n            (input)  integer
*
*   ROUTINES
*   CALLED:         FUNCTIONS
*
*                   Gammln(x)
*
*   AUTHOR:         Numerical Recipes The Art of Scientific
*                   Computing Cambridge University Press
*
*************************************************************************

      real*8 function factln(n)

      implicit none
      real*8 a,gammln
      integer n
      dimension a(100)

*  INITIALIZE THE TABLE TO NEGATIVE VALUES

      data a/100*-1./

      if (n.lt.0) pause 'negative factorial'

*  CHECK WHETHER IN RANGE OF TABLE

      if (n.le.99) then

*    IF NOT ALREADY IN THE TABLE, PUT IT IN
          if (a(n+1).lt.0) a(n+1)=gammln(n+1.)
          factln=a(n+1)

*    OUT OF RANGE OF THE TABLE
      else
          factln=gammln(n+1)
      end if

      return
      end
```

```
************************************************************************
*
*   ROUTINE:        Function
*                   GAMMLN(xx)
*
*   DESCRIPTION:    This function returns the value ln[gamma(xx)] for
*                   xx > 0.  Full accuracy is obtained for xx > 1.
*
*   ARGUMENTS:
*       xx          (input)  real
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Numerical Recipes The Art of Scientific
*                   Computing Cambridge University Press
*
************************************************************************

      real*8 function gammln(xx)

      implicit none
      integer j
      real xx
      real*8 cof(6),stp,half,one,fpf,x,tmp,ser

      data cof,stp/76.18009173d0,-86.50532033d0,24.01409822d0,-1
     +   .231739516d0,.120858003d-2,-.536382d-5,2.50662827465d0/
      data half,one,fpf/0.5d0,1.0d0,5.5d0/

      x=xx-one
      tmp=x+fpf
      tmp=(x+half)*log(tmp)-tmp
      ser=one
      do j=1,6
          x=x+one
          ser=ser+cof(j)/x
      end do

      gammln=tmp+log(stp*ser)

      return
      end
```

```
*************************************************************************
*
*   ROUTINE:       Function
*                  HIGH_TO_BANDSTOP(hpdegnum,hpnum,hpdegdenom,
*                      bs_f_center,bs_bandwidth,bsdegnum,bsnum,
*                      bsdegdenom,bsdenom)
*
*   DESCRIPTION:   An integer function which performs a high-pass to
*                  bandstop transformation on the transfer function
*                  of a linear system.
*
*   ARGUMENTS:
*       bs_bandwidth
*                  (input)  real
*                  The desired center frequency, in rad/sec, for
*                  the resulting bandstop transfer function.
*
*       bsdegdenom (output)  integer
*                  The degree of the denomenator polynomial in the
*                  bandstop transfer function.
*
*       bsdegnum   (output)  integer
*                  The degree of the numerator polynomial in the
*                  bandstop transfer function.
*
*       bsdenom    (output)  real
*                  An array containing the denominator
*                  coefficients of the bandstop transfer function.
*
*       bsnum      (output)  real
*                  An array containing the numerator coefficients
*                  of the bandstop transfer function.
*
*       bs_f_center
*                  (input)  real
*                  The desired center frequency, in rad/sec, for
*                  the resulting bandstop transfer function.
*
*       hpdegdenom (input)  integer
*                  The degree of the denominator polynomial in the
*                  system transfer function H(s).
*
*       hpdegnum   (input) integer
*                  The degree of the numerator polynomial in the
*                  system transfer function H(s).
*
*       hpdenom    (input)  real
*                  An array containing the coefficients of the
*                  denominator polynomial.
*
*       hpnum      (input)  real
*                  An array containing the coefficients of the
*                  numerator polynomial.
*
*   ROUTINES
*   CALLED:        None
*
*   AUTHOR:        Deanna L. Carroll
*                  Rt 1
*                  Lewis, KS 67552
*                  (316) 324-5338
*
*   DATE CREATED:  14July88
*************************************************************************
```

```
      integer function highpass_to_bandstop(hpdegnum,hpnum,
     +     hpdegdenom,hpdenom,bs_f_center,bs_bandwidth,bsdegnum,
     +     bsnum,bsdegdenom,bsdenom)

      integer bico,bsdegdenom,bsdegnum,deg,k,hpdegdenom,hpdegnum
      real*8 a,b,bs_bandwidth,bsdenom(0:30),bs_f_center,
     +     bsnum(0:30),c,factln,gammln,hpdenom(0:30),hpnum(0:30)

*  THIS ROUTINE WORKS FOR POLYNOMIALS OF MAX DEGREE OF 30

      a=1/bs_bandwidth
      b=(bs_f_center**2)/bs_bandwidth
      deg=max(hpdegdenom,hpdegnum)

*  CALCULATE THE NUMERATOR COEFFICIENTS AND DEGREE

      do n=0,hpdegnum,1
          c=hpnum(n)
          do k=0,n,1
              bsnum(n-2*k+deg)=c*bico(n,k)*a**(n-k)*(b**k)+
     +            bsnum(n-2*k+deg)
          end do
      end do
      bsdegnum=deg+hpdegnum

*  CALCULATE THE DENOMINATOR COEFFICIENTS AND DEGREE

      bsdegdenom=2*deg
      do n=0,hpdegdenom,1
          c=hpdenom(n)
          do k=0,n,1
              bsdenom(n-2*k+deg)=c*bico(n,k)*a**(n-k)*(b**k)+
     +            bsdenom(n-2*k+deg)
          end do
      end do

      error=0
      return
      end
```

```
*************************************************************************
*
*   ROUTINE:       Subroutine
*                  SIMPLE_PLOT(num_points,x_data,y_data,xtitle,
*                      xunits,ytitle,ytunits,title,plottype)
*
*   DESCRIPTION:   This routine generates a linear-log plot on
*                  either the 4014 Tektronix display or the HP-7475
*                  plotter.  This routine requires plotting
*                  utilities available on the VAX 11/750,
*                  EECE Dept. Kansas State University, Manhattan,
*                  KS.
*
*   ARGUMENTS:
*       title      (input)  character*40
*                  The title of the plot.
*
*       num_points
*                  (input)  integer
*                  The number of data points to be plotted.
*
*       x_data     (input)  real
*                  An array of the x coordinates of the data
*                  points.
*
*       xtitle     (input)  character*40
*                  The x axis plot label.
*
*       xunits     (input)  character*40
*                  The units of the x axis information.
*
*       y_data     (input) real
*                  An array of the y coordinates of the data
*                  points.
*
*       y_title    (input)  character*40
*                  The y axis plot label.
*
*       yunits     (input)  character*40
*                  The units of the y axis information.
*
*   ROUTINES
*   CALLED:         SUBROUTINES
*
*                   pinit(devnum,p_file,factor,size)
*
*                   pstvel(vel)
*
*                   porig(x,y)
*
*                   plogsc(data,num_points,length,first,clen,
*                       negflg)
*
*                   pscale(data,num_points,length,first,delta,
*                       divleny)
*
*                   plgaxs(xorig,yorig,title,uints,form,length,
*                       angle,first,clen)
*
*                   paxis(xorig,yorig,title,units,forlab,fortic,
*                       length,angle,first,delta,divlen)
*
*                   pplot(xcenter,ycenter,updown)
*
```

```
*
*                ptext(title)
*
*                plnlog(x_data,y_data,num_points,firlen,scntl,
*                    sybol,divleny)
*
*                pclosp
*
*   AUTHOR:      Deanna L. Carroll
*                Rt 1
*                Lewis, KS 67552
*                (316) 324-5338
*
*   DATE CREATED:  23July88
*
************************************************************************

      subroutine simple_plot(num_points,x_data,y_data,xtitle,
     +    xunits,ytitle,yunits,title,plot_type)

      implicit none
      integer devnum,dum,forlabx,forlaby,formx,formy,forticx,
     +    forticy,i,negflgx,negflgy,num_points,scntl,updown
      real anglex,angley,clenx,cleny,deltax,deltay,divlenx,
     +    divleny,factor,firdel(4),firlen(4),firstx,firsty,lengthx,
     +    lengthy,lenguu,lenstr,vel,x,xcenter,x_data(300),xorig,y,
     +    y_data(300),ycenter,yorig
      character*40 title,xtitle,xunits,ytitle,yunits,plot_type
      character*1 p_file,size,symbol

*  INITIALIZATIONS

      xorig=0
      yorig=0
      x=5.0
      y=5.0
      vel=10
      print*,'device = ?   4014 or 7475'
      read*, devnum
      p_file=' '
      factor=1
      size='A'
      lengthx=18
      forlabx=221
      forticx=2001
      anglex=0
      lengthy=12
      forlaby=120
      forticy=1001
      angley=90
      scntl=1
      symbol=' '
      formx=2011
      formy=-1011
      updown=0
      xcenter=6.0
      ycenter=14.0

*  INITIALIZE THE PLOTING DEVICE
      call pinit(devnum,p_file,factor,size)
      call pstvel(vel)
      call porig(x,y)
```

```
*  PLOT SCALE
      call plogsc(x_data,num_points,lengthx,firstx,clenx,
     +     negflgx)
      call pscale(y_data,num_points,lengthy,firsty,deltay,
     +     divleny)
      firlen(1)=firstx
      firlen(2)=clenx
      firlen(3)=firsty
      firlen(4)=deltay

*  PLOT AXES
      call plgaxs(xorig,yorig,xtitle,xunits,formx,lengthx,
     +     anglex,firstx,clenx)
      call paxis(xorig,yorig,ytitle,yunits,forlaby,forticy,
     +     lengthy,angley,firsty,deltay,divleny)

*  PLOT TITLE
      call pplot(xcenter,ycenter,updown)
      call ptext(title)

*  PLOT DATA
      call plnlog(x_data,y_data,num_points,firlen,scntl,
     +     symbol,divleny)

      call pclosp

      return
      end
```

```
************************************************************************
*
*   ROUTINE:        Mainline
*                   INIT
*
*   DESCRIPTION:    This routine assigns initial values of 0.0 to
*                   the data file "Mag.dat".  The file consists of
*                   300 real values.
*
*   ARGUMENTS:      None
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:  8Aug88
*
************************************************************************

        implicit none
        integer i
        real dbmagsc(300)

*  INITIALIZE 300 ARRAY VALUES TO 0.0

        do i=1,300
            dbmagsc(i)=0.0
        end do

*  SAVE ARRAY THE ARRAY IN A DATA FILE "MAG.DAT"

      open (unit=1,status='new',file='mag.dat',recordtype='variable',
     +   carriagecontrol='none',access='sequential')
        write (unit=1,fmt=1000) dbmagsc
        close (unit=1,status='keep')

        stop
1000    format(f8.3)
        end
```

```
************************************************************************
*
*   ROUTINE:        Mainline
*                   INVERT
*
*   DESCRIPTION:    This routine inputs 12-bit data from disk,
*                   inverts the data, and stores the results on
*                   disk.
*
*   ARGUMENTS:      None
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:  4Aug88
*
************************************************************************

        implicit none
        integer x(6000),i
        character*16 datain,dataout

*  RETRIEVE THE INPUT DATA

        print*, 'input data filename = ?'
        read*, datain
        open (unit=1,status='old',file=datain,recordtype=
     +      'variable',carriagecontrol='none',access='sequential')
        read (unit=1,fmt=1000) x
        close (unit=1,status='keep')

*  INVERT THE 12-BIT DATA

        do i=1,6000,1
            x(i)=4096-x(i)
        end do

*  SAVE THE INVERTED DATA

        print*, 'output data filename = ?'
        read*, dataout
        open (unit=1,status='new',file=dataout,recordtype=
     +      'variable',carriagecontrol='none',access='sequential')
        write (1,1000) x
        close (unit=1,status='keep')

        stop
1000    format(i12)
        end
```

```
************************************************************************
*
*   ROUTINE:       Mainline
*                  PLOT
*
*   DESCRIPTION:   This routine generates two plots on either the
*                  Tetronix 4014 display or the HP-7475 plotter.
*                  It is intended for pressure plots of 6000
*                  points.  The user enters data filenames,
*                  and scaling information.  This routine requires
*                  plot utilities available on the VAX 11/750,
*                  EECE Dept. Kansas State University, Manhattan,
*                  KS.
*
*   ARGUMENTS:     None
*
*   ROUTINES
*   CALLED:        SUBROUTINES
*
*                  pscale(data,num_points,length,first,delta,
*                      devlen)
*
*                  paxis(xorig,yorig,title,units,forlab,fortic,
*                      length,angle,first,delta,divlen)
*
*                  pstchr(width,height,slant)
*
*                  ptxtln(string,lenstr)
*
*                  pplot(xcenter,ycenter,updown)
*
*                  ptext(string)
*
*                  pline(x_data,y_data,num_points,firdel,scntl,
*                      symbol,divlenx,divleny)
*
*                  pclosp
*
*   AUTHOR:        Deanna L. Carroll
*                  Rt 1
*                  Lewis, KS 67552
*                  (316) 324-5338
*
*   DATE CREATED:  1Aug88
*
************************************************************************


      implicit none
      integer devnum,dum,forlabx,forlaby,formx,formy,forticx,fo
     +  rticy,i,negflgx,negflgy,num_points,scntl,type,updown,
     +  xp(6000)
      real anglex,angley,clenx,cleny,deltax,deltay,divlenx,divle
     +  ny,factor,firdel(4),firlen(4),firstx,firsty,lengthx,l
     +  engthy,lenguu,lenstr,vel,x,xcenter,x_data(6000),xorig,y,
     +  y_data(6000),ycenter,yorig,a,b,width,height,slant,
     +  xdum(2),ydum(2),lenstr2
      character*40 plot_title,plot_type,x_axis_title,x_axis_units,
     +  y_axis_title,y_axis_units,filename
      character*1 p_file,size,symbol
```

```
*  AXES LABELS
        plot_type='linear'
        x_axis_title='Time'
        x_axis_units='sec'
        y_axis_title='Pressure'
        y_axis_units='mmHg '

*  SIZE OF DATA FILES
        num_points=5900

*  DATA INFORMATION FOR LOWER PLOT
        print*, 'lower plot title = ?'
        read*, plot_title
        print*, 'lower data filename = ?'
        read*, filename

*  RETRIEVE DATA FOR LOWER PLOT
        open (unit=1,status='old',file=filename,recordtype=
     +    'variable',carriagecontrol='none',access='sequential')
        read (unit=1,fmt=1000) xp
        close (unit=1,status='keep')

*  SCALING INFORMATION FOR BOTH PLOTS
        print*,' The plots will be scaled using  y=a+b*x .'
        print*, 'a = ?'
        read*,a
        print*,'b = ?'
        read*,b

*  SCALE LOWER PLOT DATA
        do i=0,5900,1
            x_data(i)=(i+50)/300.0
            y_data(i)=a+b*real(xp(i+50))
        end do

*  POSITION LOWER PLOT ORIGIN
        x=7.9
        y=6.6

*  INITIALIZATIONS
        xorig=0
        yorig=0
        vel=10
        print*, 'device= ?   4014 or 7475'
        read*, devnum
        p_file=' '
        factor=1
        size='b'
        lengthx=29.25
        forlabx=221
        forticx=2001
        anglex=0
        lengthy=5
        forlaby=120
        forticy=1001
        angley=90
        scntl=1
        symbol=' '
        formx=2011
        formy=-1011
        ycenter=7
        width=.3
        height=.5
```

```
      slant=0
      updown=0
      print*,'first y-axis value = ?'
      read*,ydum(1)
      print*,'last y-axis value = ?'
      read*,ydum(2)
      call pinit(devnum,p_file,factor,size)
      call pstvel(vel)
      call porig(x,y)
      xdum(1)=0.0
      xdum(2)=20.0

*  SCALE AXES
      call pscale(xdum,2,lengthx,firstx,deltax,divlenx)
      call pscale(ydum,2,lengthy,firsty,deltay,divleny)
      firdel(1)=firstx
      firdel(2)=deltax
      firdel(3)=firsty
      firdel(4)=deltay

*  LOWER PLOT AXES
      call paxis(xorig,yorig,x_axis_title,x_axis_units,
     +  forlabx,forticx,lengthx,anglex,firstx,deltax,divlenx)
      call paxis(xorig,yorig,y_axis_title,y_axis_units,forlaby,
     +  forticy,lengthy,angley,firsty,deltay,divleny)

*  LOWER PLOT TITLE
      call pstchr(width,height,slant)
      call ptxtln(plot_title,lenstr)
      xcenter=11-lenstr/2
      call pplot(xcenter,ycenter,updown)
      call ptext(plot_title)

*  LOWER PLOT DATA
      call pline(x_data,y_data,num_points,firdel,scntl,symb
     + ol,divlenx,divleny)
      call pclosp

*  POSITION UPPER PLOT ORIGIN
      y=17.85

*  RETRIEVE DATA FOR UPPER PLOT

      print*,'upper data filename = ?'
      read*,filename
      open (unit=1,status='old',file=filename,recordtype=
     +    'variable',carriagecontrol='none',access='sequential')
      read (unit=1,fmt=1000) xp
      close (unit=1,status='keep')

*  UPPER PLOT INFORMATION
      print*, 'upper plot title = ?'
      read*, plot_title
      num_points=5900

*  SCALE UPPER PLOT DATA
      do i=0,5900,1
         x_data(i)=(i+50)/300.0
         y_data(i)=a+b*real(xp(i+50))
      end do

      call pinit(devnum,p_file,factor,size)
      call porig(x,y)
```

```
*  UPPER PLOT AXES
       call paxis(xorig,yorig,x_axis_title,x_axis_units,
     +   forlabx,forticx,lengthx,anglex,firstx,deltax,divlenx)
      call paxis(xorig,yorig,y_axis_title,y_axis_units,forlaby,
     +   forticy,lengthy,angley,firsty,deltay,divleny)

*  UPPER PLOT TITLE
      call pstchr(width,height,slant)
      call ptxtln(plot_title,lenstr2)
      xcenter=11-lenstr2/2
      call pplot(xcenter,ycenter,updown)
      call ptext(plot_title)

*  UPPER PLOT DATA
       call pline(x_data,y_data,num_points,firdel,scntl,symb
     +   ol,divlenx,divleny)

       call pclosp
       stop
1000   format(i12)
       end
```

```
************************************************************************
*
*   ROUTINE:        Mainline
*                   TRAN
*
*   DESCRIPTION:    This routine generates pulmonary transmural
*                   pressure data and saves it on disk.
*
*   ARGUMENTS:      None
*
*   ROUTINES
*   CALLED:         None
*
*   AUTHOR:         Deanna L. Carroll
*                   Rt 1
*                   Lewis, KS 67552
*                   (316) 324-5338
*
*   DATE CREATED:  22Dec88
*
************************************************************************

        implicit none

        real a1,b1,a2,b2
        integer p(6000),e(6000),t(6000),i
        character*40 pulm,esop,trans

*  RETRIEVE PULMONARY AND ESOPHAGEAL DATA

        print*,'pulmonary filename = ?'
        read*,pulm
        print*,'esophageal filename = ?'
        read*,esop
        open(unit=1,status='old',file=pulm,recordtype='variable',
     +   carriagecontrol='none',access='sequential')
        read(unit=1,fmt=1000) p
        close(unit=1,status='keep')
        open(unit=2,status='old',file=esop,recordtype='variable',
     +   carriagecontrol='none',access='sequential')
        read(unit=2,fmt=1000) e
        close(unit=2,status='keep')

*  OBTAIN SCALED PULMONARY TRANSMURAL PRESSURE

        print*,' '
        print*,' '
        print*,'to obtain the transmural pulmonary artery'
        print*,'pressure in mmHg, the data will be scaled using'
        print*,' '
        print*,'y=a1*pulm. pressure+b1-a2*esop. pressure-b2'
        print*,' '
        print*,'where a1 &b1 are scaling factors used to convert'
        print*,'pulmonary arterial pressure data to mmHg'
        print*,' '
        print*,'where a2 & b2 are scaling factors used to convert'
        print*,'esophageal pressure data to mmHg'
```

```
      print*,' '
      print*,' '
      print*,'a1 = ?'
      read*,a1
      print*,'b1 = ?'
      read*,b1
      print*,'a2 = ?'
      read*,a2
      print*,'b2 = ?'
      read*,b2

      do i=1,6000,1
          t(i)=nint(a1*real(p(i))+b1-a2*real(e(i))-b2)
      end do

*  SAVE SCALED PULMONARY TRANSMURAL DATA

      print*,'transmural filename = ?'
      read*,trans
      open(unit=3,status='new',file=trans,recordtype='variable',
     +  carriagecontrol='none',access='sequential')
      write(unit=3,fmt=1000) t
      close(unit=3,status='keep')

      stop
1000  format(i12)
      end
```

APPENDIX C

BANDSTOP FILTER COEFFICIENTS

The digital filter used was composed of a series of nine bandstop filters. The discrete-time bandstop transfer function is shown in Eq. (C1). The filter coefficients are shown in Table C.1, where f_c is the center frequency in Hz.

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}} \tag{C1}$$

Table C.1. Filter Coefficients

f_c	n	coefficients a_n	b_n
1.88	0	884898.206916424	887277.746552328
	1	-3536862.13976174	-3541604.97573160
	2	5303929.96512046	5303917.37732046
	3	-3536862.13976174	-3532119.30379188
	4	884898.206916424	882531.255080519
3.76	0	4.421932739828520E+018	4.491013503874424E+018
	1	-3.526583102227600E+019	-3.567823098124490E+019
	2	1.231573505972820E+020	1.241153584204161E+020
	3	-2.459878428142274E+020	-2.469413870291924E+020
	4	3.073487810053119E+020	3.073447895893965E+020
	5	-2.459878428142274E+020	-2.450310995192914E+020
	6	1.231573505972820E+020	1.222025319317811E+020
	7	-3.526583102227600E+019	-3.485663014327814E+019
	8	4.421932739828520E+018	4.353654234064882E+018
5.64	0	4.421932739828406E+018	4.491013503874309E+018
	1	-3.512894954203470E+019	-3.553974880380751E+019
	2	1.223401958881456E+020	1.232918455096538E+020
	3	-2.439511136295624E+020	-2.448967556552598E+020
	4	3.046358692540780E+020	3.046319025470964E+020
	5	-2.439511136295624E+020	-2.430022849408649E+020
	6	1.223401958881456E+020	1.213917107153366E+020
	7	-3.512894954203470E+019	-3.472133694326191E+019
	8	4.421932739828406E+018	4.353654234064769E+018

Table C.1. Filter Coefficients (cont.)

f_c	n	coefficients a_n	b_n
7.52	0	9.323630487103264E+019	9.391392977396066E+019
	1	-7.366583727313344E+020	-7.406701604362795E+020
	2	2.555566941512654E+021	2.564835585336098E+021
	3	-5.084117632138522E+021	-5.093322858500826E+021
	4	6.343945551983017E+021	6.343927495347709E+021
	5	-5.084117632138522E+021	-5.074897852839414E+021
	6	2.555566941512654E+021	2.546312670492620E+021
	7	-7.366583727313344E+020	-7.326611379631933E+020
	8	9.323630487103264E+019	9.256236380000240E+019
9.40	0	2.446394322247555E+020	2.460356151796724E+020
	1	-1.919309928844742E+021	-1.927519353648847E+021
	2	6.625261792705108E+021	6.644138399129102E+021
	3	-1.314143229902226E+022	-1.316013040671881E+022
	4	1.638168255088109E+022	1.638165362814802E+022
	5	-1.314143229902226E+022	-1.312271078361612E+022
	6	6.625261792705108E+021	6.606408141816637E+021
	7	-1.919309928844742E+021	-1.911123911750225E+021
	8	2.446394322247555E+020	2.432492164673807E+020
11.28	0	2.238603613183945E+019	2.261873229459802E+019
	1	-1.741137748622892E+020	-1.754694149644902E+020
	2	5.973764560114284E+020	6.004725991907302E+020
	3	-1.180643075946394E+021	-1.183695771745780E+021
	4	1.469988930552397E+021	1.469980300719851E+021
	5	-1.180643075946394E+021	-1.177583360398199E+021
	6	5.973764560114284E+020	5.942871375860273E+020
	7	-1.741137748622892E+020	-1.727651545088974E+020
	8	2.238603613183945E+019	2.215514504772650E+019
13.16	0	5.827268553476471E+018	5.912201968454789E+018
	1	-4.485859799197183E+019	-4.534807762686800E+019
	2	1.528054537357864E+020	1.539147029491860E+020
	3	-3.007207032382155E+020	-3.018087519667231E+020
	4	3.738933471039299E+020	3.738890151591072E+020
	5	-3.007207032382155E+020	-2.996291097220889E+020
	6	1.528054537357864E+020	1.516996155077414E+020
	7	-4.485859799197183E+019	-4.437266314469457E+019
	8	5.827268553476471F+018	5.743256097966286E+018

Table C.1. Filter Coefficients (cont.)

f_c	n	coefficients a_n	b_n
15.04	0	1.491780909998272E+021	1.497194540564964E+021
	1	-1.134705111882043E+022	-1.137792070189837E+022
	2	3.833336327097008E+022	3.840285058992445E+022
	3	-7.507280140826373E+022	-7.514078995423174E+022
	4	9.318955657935179E+022	9.318948856019922E+022
	5	-7.507280140826373E+022	-7.500475682108238E+022
	6	3.833336327097008E+022	3.826392923584667E+022
	7	-1.134705111882043E+022	-1.131623757695583E+022
	8	1.491780909998272E+021	1.486382014753197E+021
16.92	0	5.465340967569260E+019	5.510754173239079E+019
	1	-4.100600856172448E+020	-4.126129267393620E+020
	2	1.372356434537857E+021	1.378045355184801E+021
	3	-2.672919976257155E+021	-2.678449970159077E+021
	4	3.311953469607659E+021	3.311940725880741E+021
	5	-2.672919976257155E+021	-2.667379401618512E+021
	6	1.372356434537857E+021	1.366677437183566E+021
	7	-4.100600856172448E+020	-4.075178252318477E+020
	8	5.465340967569260E+019	5.420209805325794E+019

INVESTIGATION OF NON-PHYSIOLOGICAL PRESSURE SOURCES ASSOCIATED WITH EXERCISE DYNAMICS PRESENT IN EQUINE PULMONARY ARTERY AND ESOPHAGEAL PRESSURES RECORDED WITH A MILLAR TRANSDUCER

by

DEANNA L. CARROLL

B.S., Kansas State University, 1986

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

ABSTRACT

Equine studies investigating a proposed link between pulmonary artery hemorrhage in the horse and exercise were performed. A procedure was developed which identified the components of pulmonary artery, esophageal, and transmural pulmonary artery pressure signals by using power spectral density calculations. Comparisons of the signal components present in exercise and immediately post-exercise data demonstrated that a Millar pressure transducer, in addition to recording physiological pressures, records pressures due to non-physiological sources associated with the dynamics of exercise. The presence of the dynamic pressure components was illustrated by filtering the recorded signals. Under exercise conditions the filtering of the pulmonary artery signal resulted in a periodic waveform with reduced peaks. In addition, the filtered signal more clearly displayed the cardiac influence. The results suggest a possible correlation between equine pulmonary artery hemorrhage and exercise due to non-physiological pressure sources associated with the exercise dynamics.